10173127_Diptan_Regmi_22COM

by Student User

Submission date: 04-Sep-2020 06:48PM (UTC+0530)
Submission ID: 1379609933
File name: 10173127_Diptan_Regmi_22COM_2525_423875360.pdf (6.46M)
Word count: 652
Character count: 4387

in collaboration with

Softwarica College of IT & E-commerce | Coventry University

Module:

STW122COM – INTRODUCTION TO ALGORITHMS

Assignment Title:

INDIVIDUAL PROJECT –

HOTEL MANAGEMENT SYSTEM

Assessment Cycle

MARCH 2020

Important notes

- Please refer to the Assignment Presentation Requirements for advice on how to set out your assignment. These can be found on the *Softwarica's Moodle Course Page*.
- You are expected to use the CUHarvard referencing format on any written work. For support and advice on how this students can contact Centre for Academic Writing (CAW).
- Please notify your registry course support team and module leader for disability support.
- Any student requiring an extension should follow the university process as outlined at here
- The College cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computer. Students should therefore regularly back-up any work and are advised to save it on external media or system.
- If there are technical or performance issues that prevent students submitting Coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will be communicated via email and as a Softwarica's Moodle announcement.
- You **must** complete the **'Assessment Submission and Declaration Form'**. The form is available on *Softwarica's Moodle Course Page*.
- Please make a note of the recommended word count. You could lose marks if you write 10% more or less than this.
- You must submit a paper copy and digital copy (on disk or similarly acceptable medium). Media containing viruses, or media that cannot be run directly, will result in a fail grade being awarded for this assessment.
- All electronic media will be checked for plagiarism.

Softwarica College in collaboration with
Coventry University
Assessment Submission and Declaration Form
PLEASE COMPLETE SECTIONS IN BLOCK CAPITALS

Group work If group work ALL student names and IDs must be added below- on behalf of all members; Name..... ID..... Name..... ID..... Name..... ID..... Name..... ID..... Name..... ID.....	**Surname:** REGMI **First Name:** DIPTAN **Word Count:**
Student number (ID): 10173127	**Attempt:** FIRST ☐ RESIT ☐
Assignment Due Date: SEPTEMBER 4 2020	**Module Code:** 122COM
Programme Title: BSC(HONS) COMPUTING	
Module Title: 122COM - INTRODUCTION TO ALGOITHMS	

Name of Supervisor or Tutor (if applicable): SUJIT GYWALI	**Individual Work:** ☐	**Group Work:** ☐
Assessment Title and Type(ie essay, journal, CD, Dissertation)		

I have read the Softwarica College rules and regulations on the submission of academic work and in particular the sections concerning misconduct in assessment, including plagiarism, collusion and cheating. I certify that this assignment is the result of my ownS (or group) work and contains no unreferenced material from another source and does not contravene any part of the College's rules and regulations.

I acknowledge that in submitting this work I am declaring that I (or my group) are fit to be assessed and that a deferral may not be requested following hand in.

I confirm that an electronic version of the item to be assessed where appropriate) is available and will be made available to the College by the specified deadline via Moodle.

In respect of group assignments, the submission of this work is made on the basis that all group members are jointly and severally responsible for the work presented for assessment and that by handing in this item for assessment, all group members acknowledge and confirm the statements above and that ALL student names and ID numbers for the group are listed.

Student(s) Signature:	**College Stamp**

Contents

INTRODUCTION

In this digitalize world, many hotels are keeping their records in paper and that makes them difficult to see and maintain the data and records. So, I built a simple desktop application which will be effective for medium to small hotels. This Hotel management system have a login which will be created by admin and only admin can add, update and delete the details of hotel rooms and menu items available in hotel and they created other user to use the application which helps in operating the system securely. With all the data saved in database it makes the hotel easier to view the details of guest and generate bills. The customer data saved while booking can be used to promote the business of the hotel and the billings can be used for accounting easily. With the use of Hotel Management System it makes simpler, easier and efficient to operate the hotel.

This system is built on python 3.8 with Tkinter GUI in the interface and MySQL database to store the data. The concept of Object Oriented Programming is used to write the code which provides modularity for easier, reuse of code through inheritance, flexibility through polymorphism and effective problem solving. The code is written with the editor PyCharm Community Edition and database is created in MySQL Workbench.

The fully automated Hotel Management System has following features:

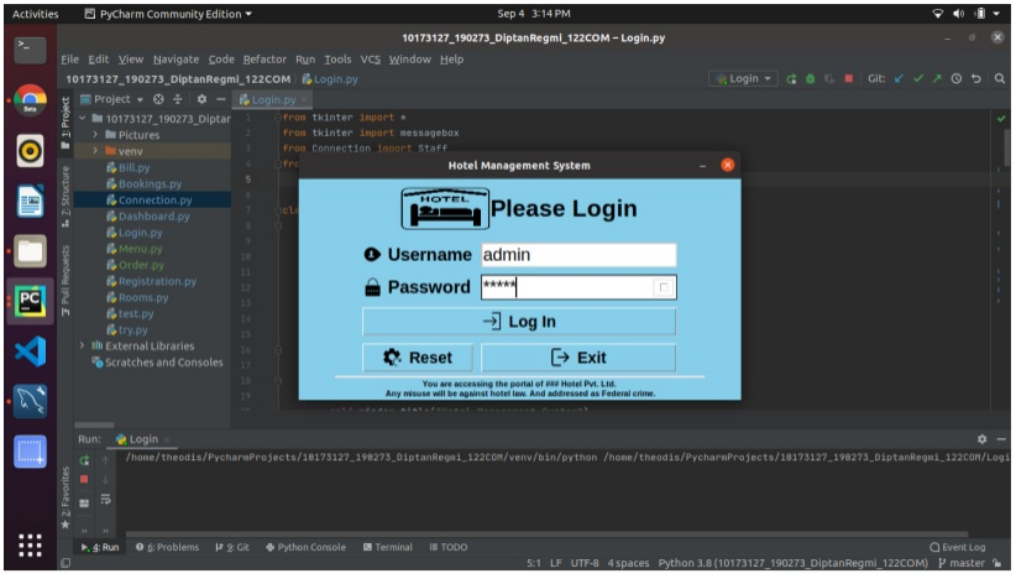
- Different level of accessibility for admin and users.
- Admin can manages hotel rooms and hotel menu.
- Only Admin can add new user to use the application.
- Both user and admin can add new bookings and add orders maid by customer and generate bills of the customer staying in the hotel.
- Staff Data, Booking Data, Room Data, Menu Data, Orders Data, and Payment Info are stored separately in MySQL database.
- All operations are validate and shows proper error messages.
- Unit testing of different function and manually tested the whole program for quality assurance.
- Quick Sort and Binary Search Algorithm is implemented in the program.

Github Repository:

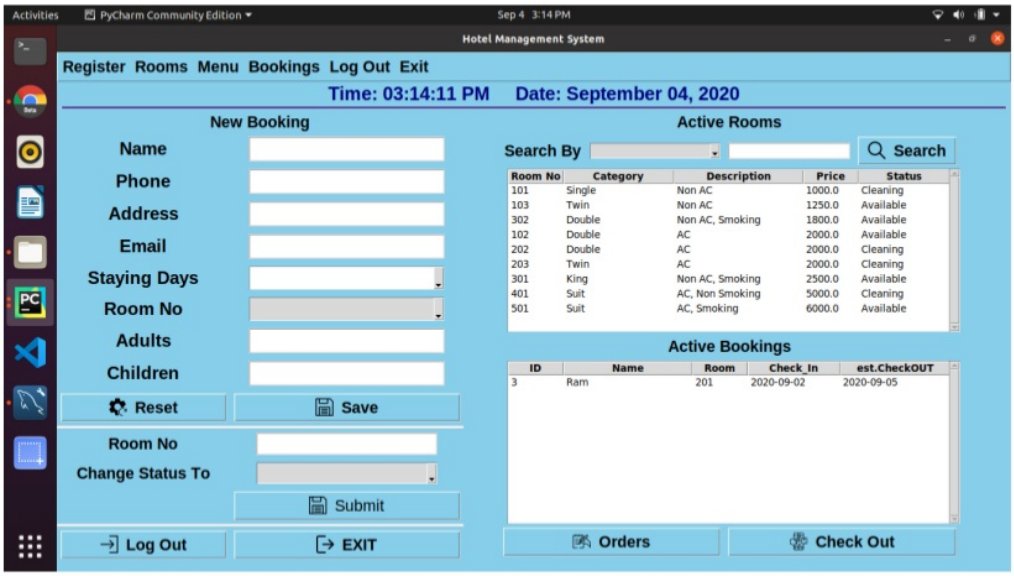
https://github.com/softwarica-github/final-assignment-rdiptan

GUI

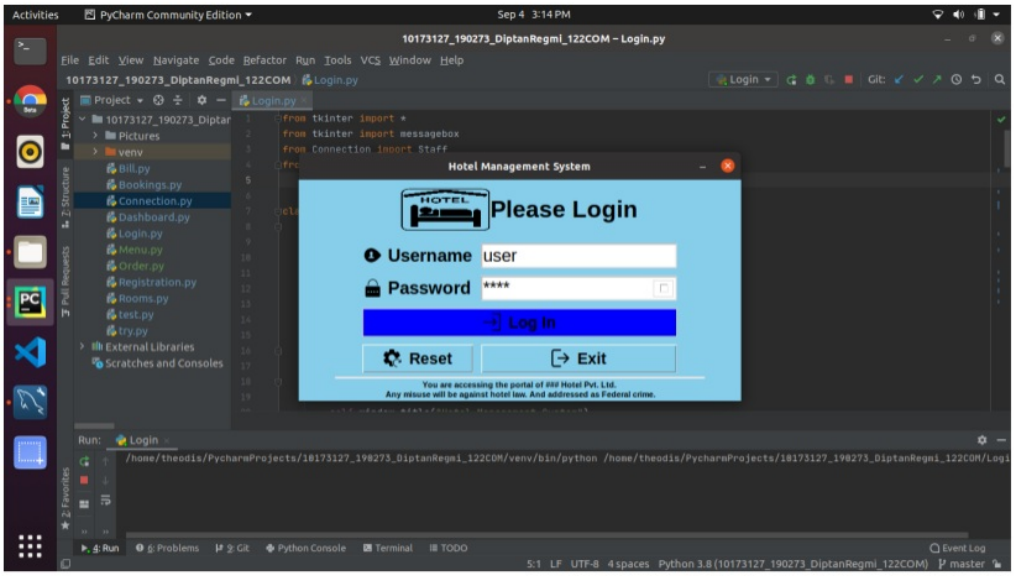
Only main pages are shown in these screenshots.



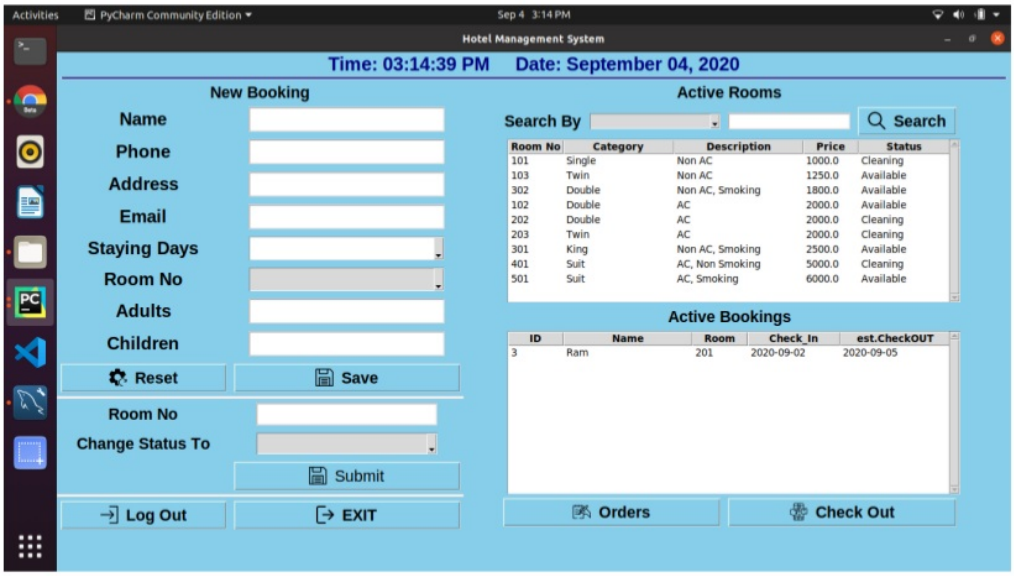
Login Page



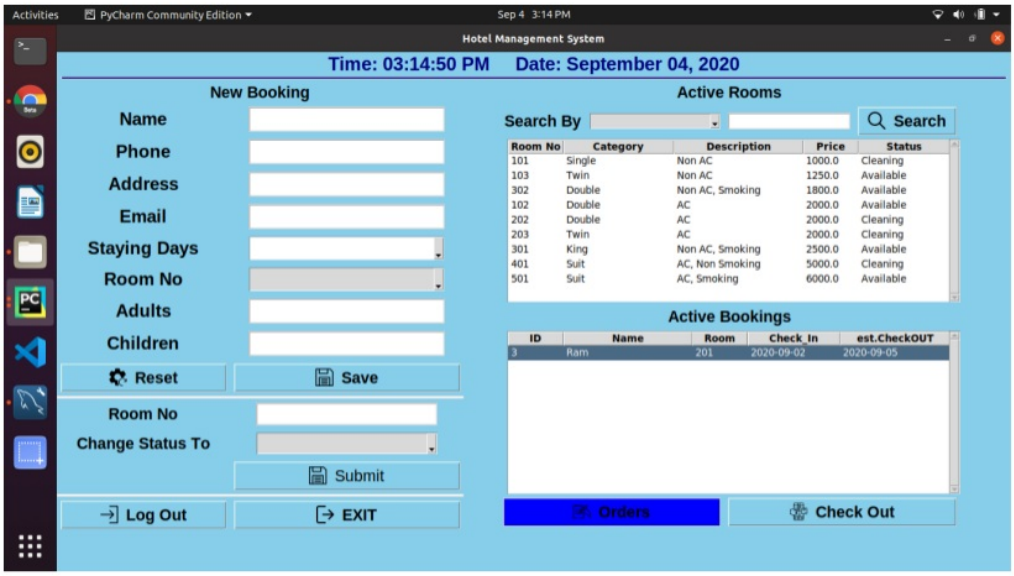
Logged In as Admin



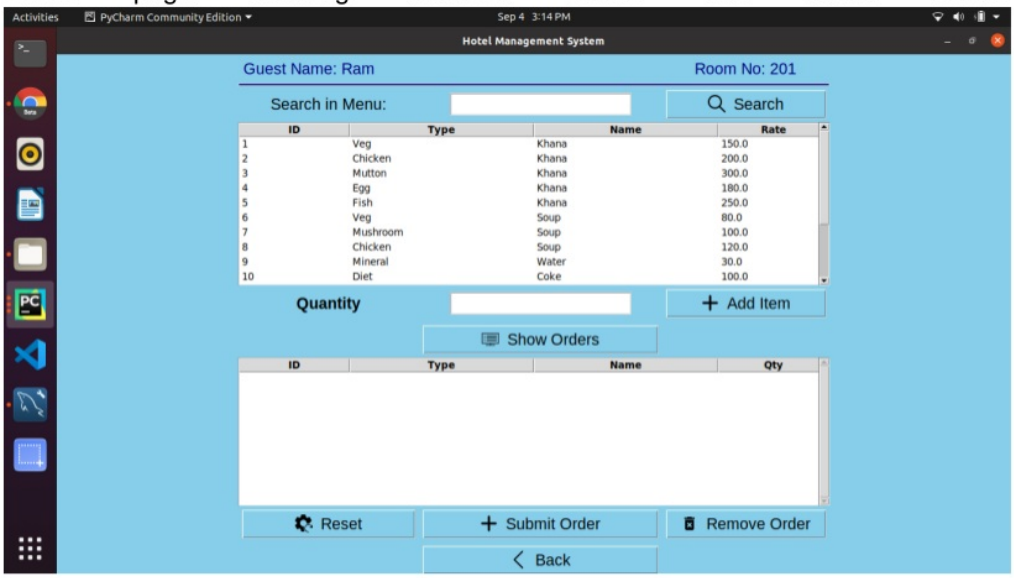
Login Page



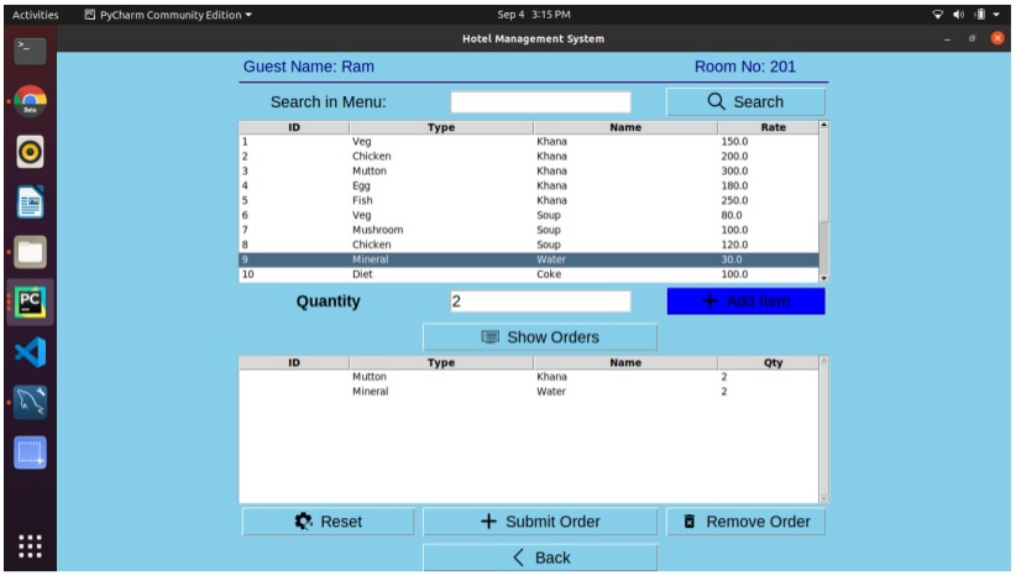
Logged in as User



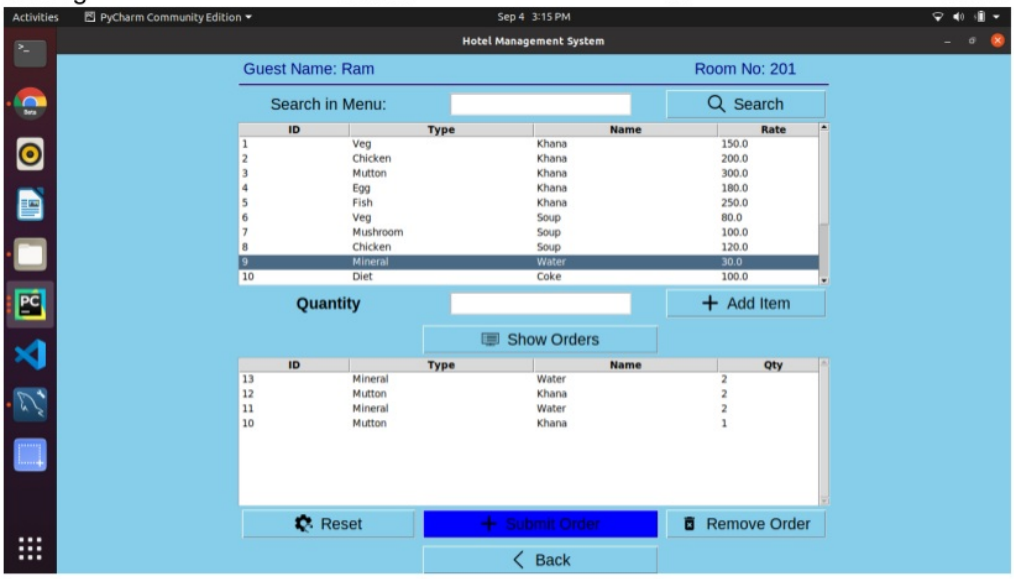
Home page showing available rooms and active orders



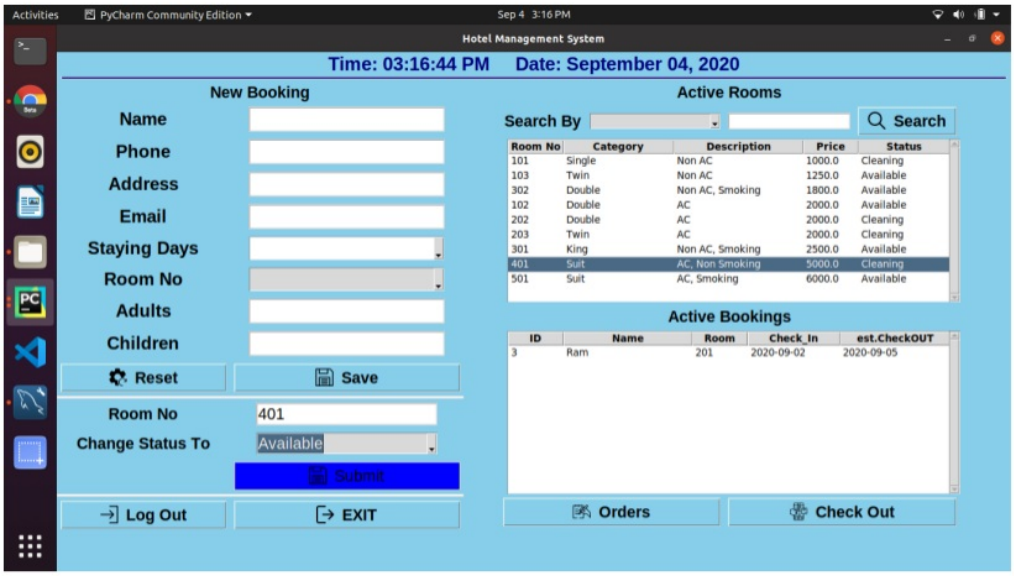
Food Ordering Page



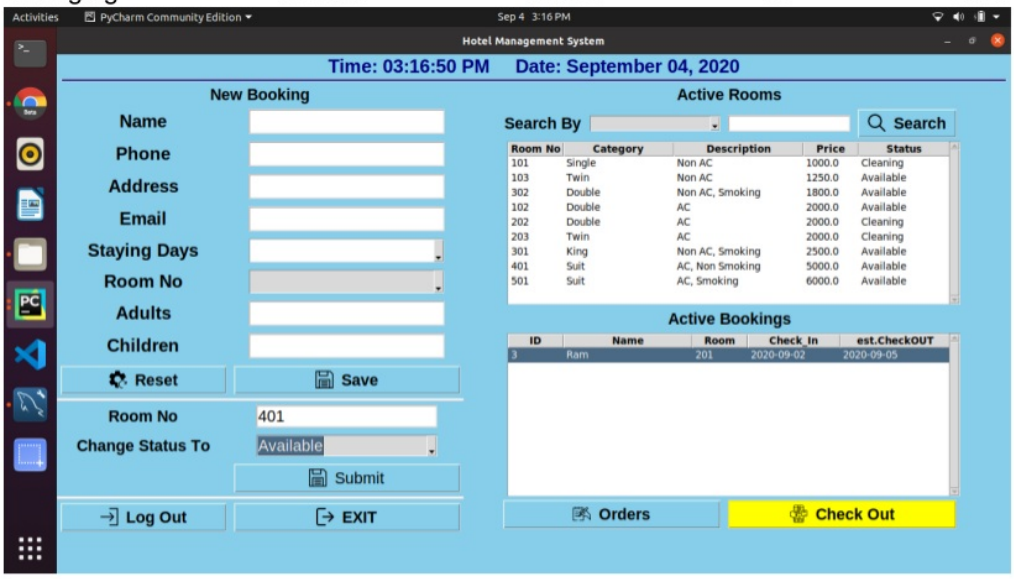
Adding foods to order



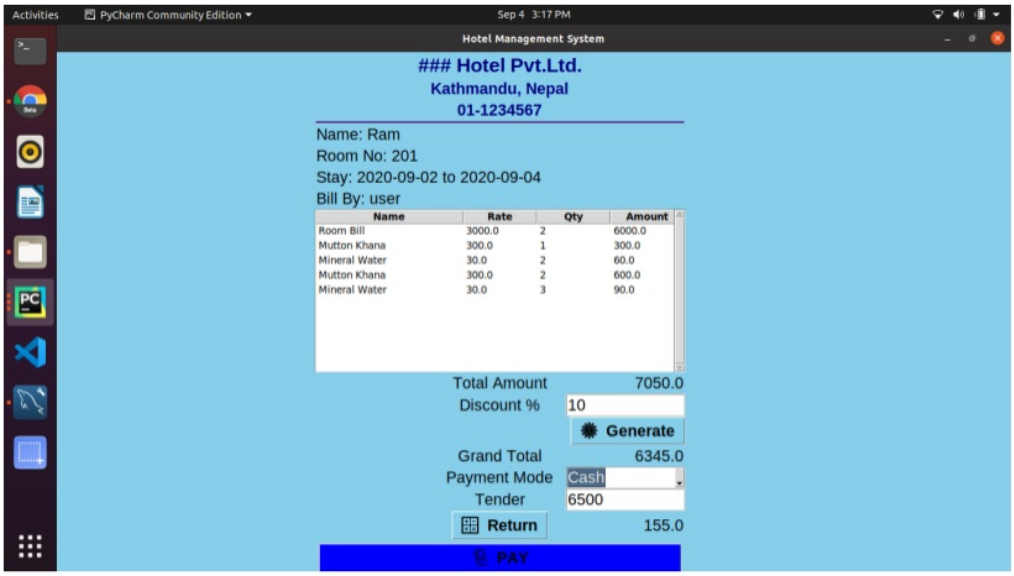
Submit the ordered food items



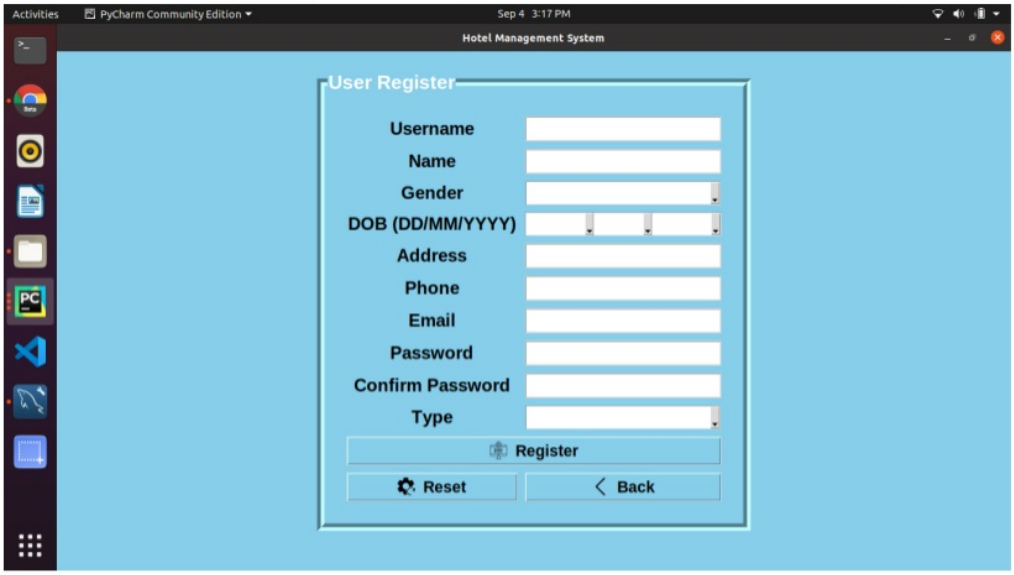
Changing current status of room



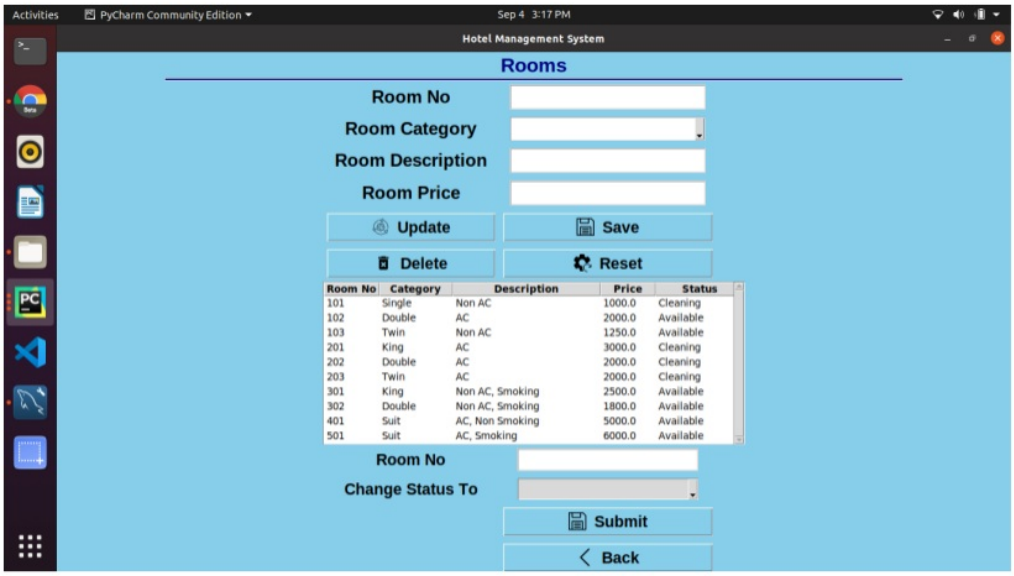
Check Out



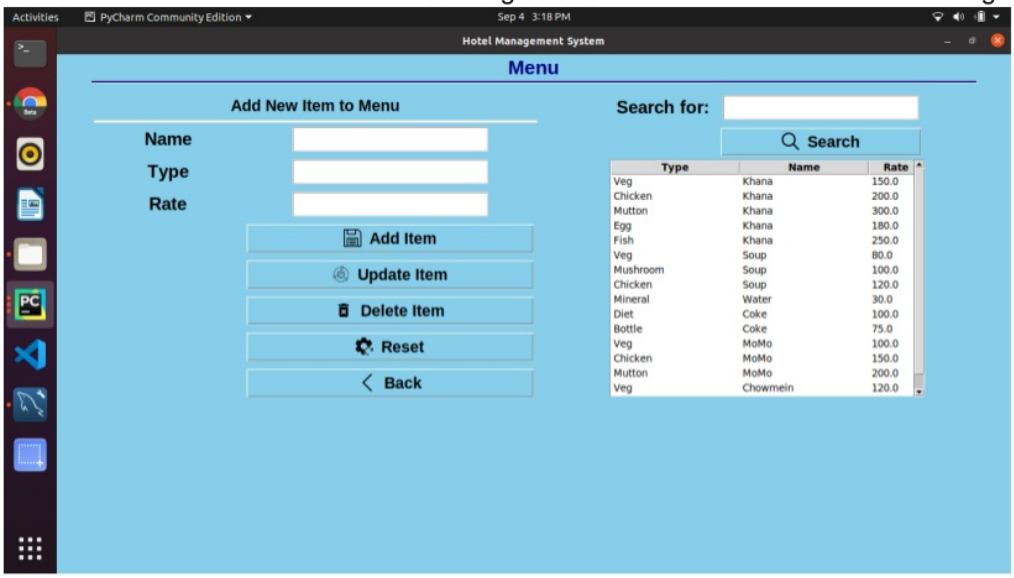
Bill Generated for selected customerU



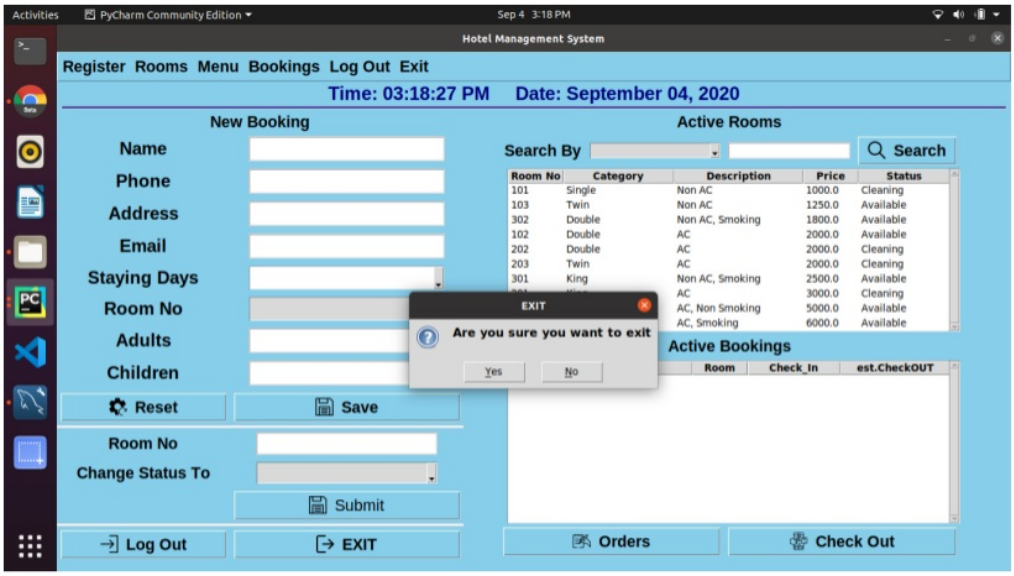
User Registration Page



Room Management Page

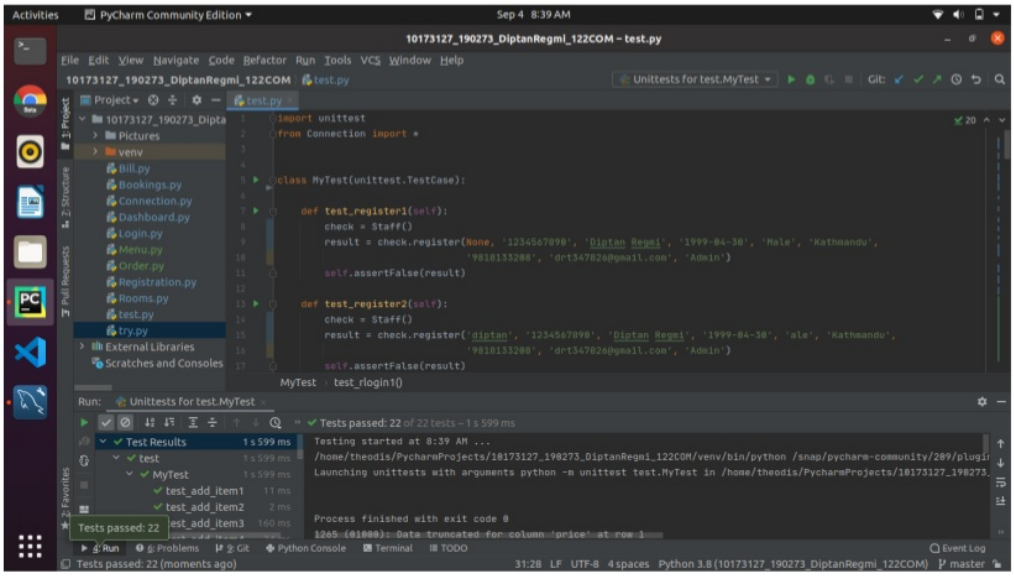


Menu management page



Quit Programme

UNIT TESTING



QUICK SORT AND BINARY SEARCH CODE

```
    def on_back_click(self):
        """takes back to hotel management system homepage"""
        self.window.destroy()

    def search_user(self, username):
        """checks for username is already present or not"""
        reg = Staff()
        data = reg.check_username()
        records = self.quick_sort(data)
        result = self.binary_search(records, username)
        if result != -1:
            return True
        else:
            return False

    def quick_sort(self, my_list):
        """Algorithm to sort the list of generated array for searching"""
        less = []
        equal = []
        greater = []

        if len(my_list) > 1:
            pivot = my_list[0]
            for x in my_list:
                if x < pivot:
                    less.append(x)
                elif x == pivot:
                    equal.append(x)
                elif x > pivot:
                    greater.append(x)
            return self.quick_sort(less) + equal + self.quick_sort(greater)

        else:
            return my_list

    def binary_search(self, my_list, key):
        """binary search algorithm"""
        start = 0
        end = len(my_list) - 1
        while start <= end:
            mid = (start + end) // 2
            if my_list[mid][0] == key:
                return mid
            elif my_list[mid][0] > key:
                end = mid - 1
            else:
                start = mid + 1
        return -1
```

CODE SCREENSHOT

```
from tkinter import *
from tkinter import messagebox
from Connection import Staff
from Dashboard import Dashboard


class Login:
    """
        This class provides login form to use the programme.

        Methods:
            on_login_click()
            on_reset_lick()
            show()

        """

    def __init__(self, window):
        self.window = window
        self.window.title("Hotel Management System")
        self.window.geometry("600x300+400+200")

        self.window.configure(bg="sky blue")

        self.window.resizable(0, 0)

        self.un = StringVar()
        self.pw = StringVar()

        # images
        canvas = Canvas()
        self.bg_entry = PhotoImage(file="Pictures/hotel2.png")
        new_image = self.bg_entry.subsample(5, 10)
        canvas.create_image(0, 0, image=new_image, anchor="nw")
        self.canvas_image = new_image

        self.bg_uname = PhotoImage(file="Pictures/uname.png")
        self.bg_pass = PhotoImage(file="Pictures/password.png")
        self.bg_log = PhotoImage(file="Pictures/logout.png")
        self.bg_reset = PhotoImage(file="Pictures/reset.png")
        self.bg_exit = PhotoImage(file="Pictures/exit.png")

        # title
        self.title0 = Label(self.window, text="Please Login", image=self.canvas_image, compound=LEFT,
                            font=('Arial', 24, 'bold'), bg="sky blue")
        self.title0.pack(pady=10)
```

```
    def on_login_click(self):
        """check for if input fields are empty or not then verify the user id and password
              and then logged into the system"""
        username = self.un.get()
        password = self.pw.get()
        valid = Staff()
        if username == "" or password == "":
            messagebox.showerror("Error", "Enter all values")
        else:
            usr = valid.login(username, password)
            if len(usr) > 0:
                self.window.destroy()
                Dashboard(usr[0])
            else:
                messagebox.showerror('Error', 'Wrong id or password')

    def on_reset_click(self):
        """resets the input values to blank"""
        self.entry_un.delete(0, END)
        self.entry_pw.delete(0, END)

    def show(self):
        """shows the password if checkbutton is clicked"""
        if self.check.get() == 0:
            self.entry_pw["show"] = "*"
        else:
            self.entry_pw["show"] = ""


def main():
    wn = Tk()
    Login(wn)
    wn.mainloop()


if __name__ == "__main__":
    main()
```

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import time
from datetime import datetime, timedelta
from Registration import Register
from Rooms import Rooms
from Bookings import Bookings
from Connection import Room
from Connection import Booking
from Bill import BillView
from Order import OrderView
from Menu import ItemView


class Dashboard:
    """
        This class works with new booking and display available rooms, active booking and acts as homepage to system.

        Methods:
            date_time()
            combo_rooms()
            on_room_search()
            show_room_tree()
            send_room()
            show_book_tree()
            on_save_click()
            on_reset_click()
            on_room_change()
            on_order()
            on_check_out()
            open_login()
            open_register()
            open_rooms()
            open_booking()
            open_menu()
            exit_handler()

        """

    def __init__(self, user):
        self.window = Tk()
        self.window.title("Hotel Management System")
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")

        self.bg_save = PhotoImage(file="Pictures/save.png")
        self.bg_reset = PhotoImage(file="Pictures/reset.png")
        self.bg_log = PhotoImage(file="Pictures/logout.png")
        self.bg_exit = PhotoImage(file="Pictures/exit.png")
        self.bg_ser = PhotoImage(file="Pictures/search.png")
        self.bg_order = PhotoImage(file="Pictures/order.png")
        self.bg_out = PhotoImage(file="Pictures/check.png")
```

```
        self.user = user

        # runs if login is made by admin
        if self.user[-1] == 'Admin':
            menubar = Menu(self.window, bg="sky blue")
            menubar.add_command(label="Register", activebackground="blue", font=("arial", 16, "bold"),
                                command=self.open_register)
            menubar.add_command(label="Rooms", activebackground="blue", font=("arial", 16, "bold"),
                                command=self.open_rooms)
            menubar.add_command(label="Menu", activebackground="blue", font=("arial", 16, "bold"),
                                command=self.open_menu)
            menubar.add_command(label="Bookings", activebackground="blue", font=("arial", 16, "bold"),
                                command=self.open_booking)
            menubar.add_command(label="Log Out", activebackground="yellow", font=("arial", 16, "bold"),
                                command=self.open_login)
            menubar.add_command(label="Exit", activebackground="red", font=("arial", 16, "bold"),
                                command=self.exit_handler)
            self.window.config(menu=menubar)

        # date and time
        self.lblInfo = Label(self.window, font=('arial', 19, 'bold'), bg='sky blue', fg="navy blue")
        self.lblInfo.pack()
        self.date_time()

        self.line = Canvas(self.window, width=1280, height=2, bg="navy blue").pack()

        # frames
        self.frame1 = Frame(self.window, bg="sky blue")
        self.frame1.place(x=0, y=40)
        self.frame2 = Frame(self.window, bg="sky blue")
        self.frame2.place(x=600, y=40)

        # title
        self.label_book = Label(self.frame1, text="New Booking", font=("arial", 16, "bold"), fg="#000000",
                                bg="sky blue")
        self.label_book.grid(row=0, column=0, columnspan=2)

        # Customer Details
        self.label_name = Label(self.frame1, text="Name", font=("arial", 18, "bold"), fg="#000000", bg="sky blue")
        self.label_name.grid(row=1, column=0, padx=5, pady=5)
        self.entry_name = Entry(self.frame1, font=("arial", 18))
        self.entry_name.grid(row=1, column=1, padx=5, pady=5)
```

```
def date_time(self):
    """display current date and time"""
    d = datetime.now()
    today = '{:%B %d, %Y}'.format(d)
    my_time = time.strftime('%I:%M:%S %p')
    self.lblInfo.config(text=('Time: ' + my_time + '     Date: ' + today))
    self.lblInfo.after(200, self.date_time)

def combo_rooms(self):
    """returns available rooms to combobox"""
    ret = Room()
    data = ret.available_rooms()
    self.combo_room['values'] = data

def on_room_search(self):
    """search for rooms and display in room's treeview"""
    category = self.combo_search1.get()
    value = self.entry_search1.get()
    if category == "" or value == "":
        messagebox.showerror("Error", "Enter all the values for searching")
    else:
        self.room_tree.delete(*self.room_tree.get_children())
        ret = Room()
        data = ret.search_rooms(category, value)
        for i in data:
            self.room_tree.insert("", "end", text=i[0], values=i)

def show_room_tree(self):
    """display rooms in treeview"""
    self.room_tree.delete(*self.room_tree.get_children())
    ret = Room()
    data = ret.show_rooms_stat()
    for i in data:
        self.room_tree.insert("", "end", text=i[0], values=i)
    self.room_tree.bind("<Double-1>", self.send_room)

def send_room(self, event):
    """returns room no to room entry box"""
    selected_item = self.room_tree.selection()[0]
    room_data = self.room_tree.item(selected_item, 'values')
    self.entry_room1.delete(0, END)
    self.entry_room1.insert(0, room_data[0])
```

```
def on_save_click(self):
    """verify for data given in entrybox and save them in database"""
    name = self.entry_name.get()
    phone = self.entry_phone.get()
    address = self.entry_address.get()
    email = self.entry_email.get()
    staying = self.combo_stay.get()
    room = self.combo_room.get()
    adults = self.entry_adult.get()
    children = self.entry_child.get()
    if not staying.isnumeric() or not adults.isnumeric() or not room.isnumeric():
        messagebox.showerror("Error", "Enter Number for adults and children and staying days")
    else:
        check_in = datetime.now().strftime('%Y/%m/%d')
        stay = datetime.now() + timedelta(days=int(staying))
        check_out = '{:%Y/%m/%d}'.format(stay)
        status = "Occupied"

        save = Booking()
        stat = Room()

        if name == "" or room == "":
            messagebox.showerror("Error", "Name and Room No is must")

        else:
            if save.new_booking(name, phone, address, email, check_in, check_out, room, adults, children):
                stat.change_status(status, room)
                a = messagebox.showinfo('Success', 'Room Booked successfully')
                if a == 'ok':
                    self.on_reset_click()
                    self.show_room_tree()
                    self.show_book_tree()
                    self.combo_rooms()
            else:
                messagebox.showerror('Booking Failed', 'Please Try Again')
```

```
def on_order(self):
    """opens the order page for selected customer"""
    try:
        selected_item = self.book_tree.selection()[0]
        book_data = self.book_tree.item(selected_item, 'values')
        booking_id = book_data[0]
        cus_name = book_data[1]
        cus_room = book_data[2]
        OrderView(booking_id, cus_name, cus_room)
    except IndexError:
        messagebox.showerror("Error", "Select the customer from active bookings")

def on_check_out(self):
    """checkout of selected customer and open billing page"""
    try:
        user = self.user[0]
        check_out = datetime.now().strftime('%Y/%m/%d')
        selected_item = self.book_tree.selection()[0]
        book_data = self.book_tree.item(selected_item, 'values')
        booking_id = book_data[0]
        up = Booking()
        up.check_out(check_out, booking_id)
        BillView(booking_id, user)
    except IndexError:
        messagebox.showerror("Check Out Error", "Select the customer from active bookings")
```

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from Connection import Item
from Connection import Order


class OrderView:
    """
        This class present food orders to customers staying in hotel.

        Methods:
            add_items_in_list()
            submit_order()
            reset_tree_view()
            show_item_tree()
            ser_item()
            show_orders()
            delete_orders()

        """

    def __init__(self, cus_id, cus_name, cus_room):
        self.window = Toplevel()
        self.window.title("Hotel Management System")
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")

        self.bg_search = PhotoImage(file="Pictures/search.png")
        self.bg_save = PhotoImage(file="Pictures/save.png")
        self.bg_show = PhotoImage(file="Pictures/show.png")
        self.bg_add = PhotoImage(file="Pictures/add.png")
        self.bg_reset = PhotoImage(file="Pictures/reset.png")
        self.bg_del = PhotoImage(file="Pictures/delete.png")
        self.bg_back = PhotoImage(file="Pictures/back.png")

        self.cus_id = cus_id
        self.cus_name = cus_name
        self.cus_room = cus_room

        self.frame2 = Frame(self.window, bg="sky blue")
        self.frame2.pack()

        self.label_name = Label(self.frame2, text=("Guest Name: " + cus_name), font=("arial", 16),
                                fg="navy blue", bg="sky blue")
        self.label_name.grid(row=0, column=0, columnspan=2, padx=5, pady=5, sticky='w')
        self.label_room = Label(self.frame2, text=("Room No: " + cus_room), font=("arial", 16),
                                fg="navy blue", bg="sky blue")
        self.label_room.grid(row=0, column=2, padx=5, pady=5)

        self.line = Canvas(self.frame2, width=800, height=2, bg="navy blue").grid(row=1, columnspan=3)
```

```
from tkinter import *
from tkinter import ttk
from Connection import Bill
from Connection import Booking
from Connection import Room
from tkinter import messagebox


class BillView:
    """
            This class process billing of the customer.

            Methods:
                on_gen_click()
                on_return_click()
                order_bill()
                on_pay_click()

            """
    def __init__(self, cus_id, user):
        self.window = Toplevel()
        self.window.title("Hotel Management System")
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")

        self.bg_pay = PhotoImage(file="Pictures/money.png")
        self.bg_gen = PhotoImage(file="Pictures/generate.png")
        self.bg_calc = PhotoImage(file="Pictures/calculate.png")

        self.id = cus_id
        self.user = user

        # customer stay details
        ret = Bill()
        val = ret.room_bill(self.id)
        data = val[0]

        # calculate hotel stay days from checked in and checking out date
        self.room_no = data[3]
        self.days = abs(data[1]-data[2]).days
        self.tot = float(data[4])*int(self.days)

        # calculating amount owe to hotel
        global total
        global total_bill
        ret = Bill()
        order_amt = ret.total_order(self.id)
        for i in order_amt:
            total = i[0]
        try:
            total_bill = self.tot + total
        except TypeError:
            total_bill = self.tot
```

```
    def order_bill(self):
        """show bills details in treeview"""
        ret = Bill()
        data = ret.order_bill(self.id)
        for i in data:
            self.bill_tree.insert("", "end", text=i[0], value=(i[0], i[1], i[2], i[3]))

    def on_pay_click(self):
        """save payment details to database"""
        booking_id = self.id
        discount = self.entry_discount.get()
        paid_amount = self.grand_total.cget("text")
        payment_type = self.combo_pay.get()
        billed_by = self.user
        status = "Cleaning"
        if paid_amount == "":
            messagebox.showerror('Payment Failed', 'Please add a discount value and press generate button\n'
                                                   'if no discount is given enter 0')
        elif payment_type == "":
            messagebox.showerror('Payment Failed', 'Please select payment mode')
        else:
            room = self.room_no
            save = Bill()
            if save.save_payment(booking_id, discount, paid_amount, payment_type, billed_by):
                a = messagebox.showinfo('Success', 'Bill Paid successfully')
                up = Booking()
                up.booking_stat(booking_id)
                stat = Room()
                stat.change_status(status, room)
                if a == 'ok':
                    self.window.destroy()
            else:
                messagebox.showerror('Payment Failed', 'Please Try Again')
```

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from Connection import Staff


class Register:
    """
        This class provides a form to register new users.

        Methods:
            on_register_click()
            on_reset_click()
            on_back_click()
            search_user()
            quick_sort()
            binary_search()

        """
    def __init__(self, window):
        self.window = window
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")

        self.bg_reg = PhotoImage(file="Pictures/reg.png")
        self.bg_reset = PhotoImage(file="Pictures/reset.png")
        self.bg_back = PhotoImage(file="Pictures/back.png")

        # label frame
        self.frame = LabelFrame(self.window, bg="sky blue", fg="white", text="User Register",
                                font=("arial", 20, "bold"), bd=10, padx=25, pady=25)
        self.frame.pack(padx=25, pady=25)

        # username
        self.label_un = Label(self.frame, text="Username", font=("arial", 18, "bold"), fg="#000000", bg="sky blue")
        self.label_un.grid(row=0, column=0, padx=5, pady=5)
        self.entry_un = Entry(self.frame, font=("arial", 18))
        self.entry_un.grid(row=0, column=1, padx=5, pady=5)
```

```
    def on_back_click(self):
        """takes back to hotel management system homepage"""
        self.window.destroy()

    def search_user(self, username):
        """checks for username is already present or not"""
        reg = Staff()
        data = reg.check_username()
        records = self.quick_sort(data)
        result = self.binary_search(records, username)
        if result != -1:
            return True
        else:
            return False

    def quick_sort(self, my_list):
        """Algorithm to sort the list of generated array for searching"""
        less = []
        equal = []
        greater = []

        if len(my_list) > 1:
            pivot = my_list[0]
            for x in my_list:
                if x < pivot:
                    less.append(x)
                elif x == pivot:
                    equal.append(x)
                elif x > pivot:
                    greater.append(x)
            return self.quick_sort(less) + equal + self.quick_sort(greater)

        else:
            return my_list

    def binary_search(self, my_list, key):
        """binary search algorithm"""
        start = 0
        end = len(my_list) - 1
        while start <= end:
            mid = (start + end) // 2
            if my_list[mid][0] == key:
                return mid
            elif my_list[mid][0] > key:
                end = mid - 1
            else:
                start = mid + 1
        return -1
```

```
class Rooms:
    """
            This class manages rooms of the hotel.

            Methods:
                show_room_tree()
                select_item()
                on_save_click()
                on_update_click()
                on_reset_click()
                on_submit_click()
                on_delete_click()

            """

    def __init__(self, window):
        self.window = window
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")

        self.bg_save = PhotoImage(file="Pictures/save.png")
        self.bg_update = PhotoImage(file="Pictures/update.png")
        self.bg_del = PhotoImage(file="Pictures/delete.png")
        self.bg_reset = PhotoImage(file="Pictures/reset.png")
        self.bg_back = PhotoImage(file="Pictures/back.png")

        self.label_number = Label(self.window, text="Rooms", font=("arial", 20, "bold"), fg="navy blue", bg="sky blue")
        self.label_number.pack()

        self.line = Canvas(self.window, width=1000, height=2, bg="navy blue").pack()
```

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from Connection import Item


class ItemView:
    """
            This class manages menu of the hotel.

            Methods:
                add_item()
                del_item()
                update_item()
                show_item_tree()
                select_item()
                reset_btn()
                ser_item()

            """

    def __init__(self, window):
        self.window = window
        self.window.title("Hotel Management System")
        self.window.geometry("1366x768+0+0")
        self.window.configure(bg="sky blue")
```

```python
import mysql.connector


class Connection:
    """
            This class connects python file with database.

            Methods:
                iud()
                insert_with_id_return()
                show_data()
                search_dat()

            """

    def __init__(self):
        self.my_connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='regmi321',
            auth_plugin='mysql_native_password',
            database='hotel'
        )
        self.my_cursor = self.my_connection.cursor()

    def iud(self, qry, values):
        """insert, update and delete data in database tables"""
        try:
            self.my_cursor.execute(qry, values)
            self.my_connection.commit()
            return True
        except Exception as e:
            print(e)
            return False

    def insert_with_id_return(self, qry, values):
        """insert data to database and return primary key"""
        try:
            self.my_cursor.execute(qry, values)
            self.my_connection.commit()
            self.my_connection.close()
            return self.my_cursor.lastrowid
        except Exception as e:
            print(e)
            return 0

    def show_data(self, qry):
        """fetch data from database table"""
        data = []
        try:
            self.my_cursor.execute(qry)
            data = self.my_cursor.fetchall()
            self.my_connection.close()
            return data
        except Exception as e:
            print(e)
            return data

    def search_data(self, qry, val):
        """search for data in database"""
        data = []
        try:
            self.my_cursor.execute(qry, val)
            data = self.my_cursor.fetchall()
            self.my_connection.close()
            return data
        except Exception as e:
            print(e)
            return data
```

```python
class Staff(Connection):
    """
        This class works with login and registration of user for application.

        Methods:
            check_username()
            register()
            login()

        """

    def __init__(self):
        super().__init__()

    def check_username(self):
        """fetch usernames from staff table"""
        qry = "SELECT username FROM staff"
        occ_user = self.show_data(qry)
        return occ_user

    def register(self, username, password, name, dob, gender, address, mobile, email, type_):
        """insert data to staff table"""
        qry = "INSERT into staff(username, password, name, dob, gender, address, mobile, email, type) " \
              "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
        values = (username, password, name, dob, gender, address, mobile, email, type_)
        return self.iud(qry, values)

    def login(self, username, password):
        """verify username and password for login"""
        qry = "SELECT * FROM staff WHERE username = %s AND password = %s"
        values = (username, password)
        user = self.search_data(qry, values)
        return user
```

```python
class Room(Connection):
    """
        This class works with rooms for hotel.

        Methods:
            create_rooms()
            update_rooms()
            show_rooms()
            show_room_stat()
            available_rooms()
            change_status()
            search_rooms()
            delete_rooms()

        """

    def __init__(self):
        super().__init__()

    def create_rooms(self, room_no, room_category, room_description, price):
        """save new rooms data to database"""
        qry = "INSERT INTO rooms(room_no, room_category, room_description, price)" \
              " VALUES(%s, %s, %s, %s)"
        values = (room_no, room_category, room_description, price)
        return self.iud(qry, values)

    def update_rooms(self, room_category, room_description, price, room_no):
        """update room details in database"""
        qry = "UPDATE rooms SET room_category = %s, room_description = %s, price = %s WHERE room_no = %s"
        values = (room_category, room_description, price, room_no)
        return self.iud(qry, values)

    def show_rooms(self):
        """returns all rooms"""
        qry = "SELECT * FROM rooms"
        room = self.show_data(qry)
        return room

    def show_rooms_stat(self):
        """return rooms with room status available and cleaning"""
        qry = "SELECT * FROM rooms WHERE room_status = %s OR room_status = %s order by price"
        values = ("Available", "Cleaning")
        room = self.search_data(qry, values)
        return room

    def available_rooms(self):
        """returns available rooms"""
        qry = "SELECT room_no FROM rooms where room_status = %s"
        values = ("Available",)
        room = self.search_data(qry, values)
        return room

    def change_status(self, status, room):
        """update room status"""
        qry = "UPDATE rooms SET room_status = %s WHERE room_no = %s"
        values = (status, room)
        return self.iud(qry, values)

    def search_rooms(self, category, value):
        """search for rooms"""
        qry = " SELECT * FROM rooms WHERE " + category + " LIKE %s AND room_no in (SELECT room_no from rooms where " \
                                                         "room_status = 'Available' OR room_status = 'Cleaning')"
        values = ("%" + value + "%",)
        room = self.search_data(qry, values)
        return room

    def delete_rooms(self, room):
        """delete specific room"""
        qry = "DELETE FROM rooms WHERE room_no = %s"
        values = (room,)
        return self.iud(qry, values)
```

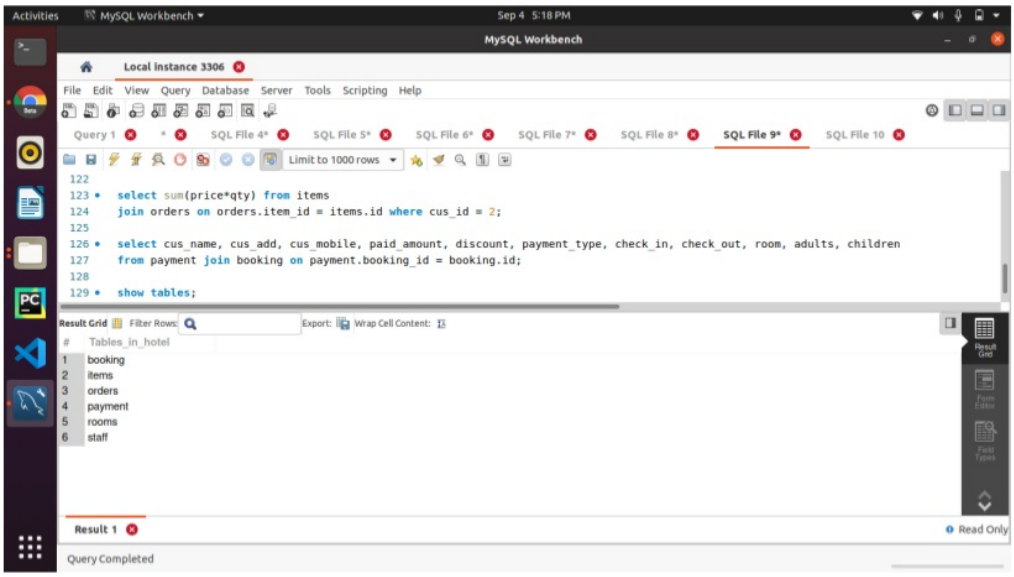
```python
class Booking(Connection):
    """
            This class works with booking of hotel's customer.

            Methods:
                new_booking()
                show_active_booking()
                check_out()
                booking_stat()

            """

    def __init__(self):
        super().__init__()

    def new_booking(self, name, phone, address, email, check_in, check_out, room, adults, children):
        """add new customer details"""
        qry = """INSERT INTO booking
        (cus_name, cus_mobile, cus_add, cus_email, check_in, check_out, room, adults, children)
        VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s)"""
        values = (name, phone, address, email, check_in, check_out, room, adults, children)
        return self.iud(qry, values)

    def show_active_booking(self):
        """returns the customer staying in hotel currently"""
        qry = "SELECT id, cus_name, room, check_in, check_out FROM booking where booking_status=%s order by check_out"
        values = ("Staying",)
        return self.search_data(qry, values)

    def check_out(self, out, id_):
        """change check_out date"""
        qry = "UPDATE booking SET check_out=%s where id=%s"
        values = (out, id_)
        return self.iud(qry, values)

    def booking_stat(self, id_):
        """on checking out of the hotel change status of customer to checked_out"""
        qry = "UPDATE booking SET booking_status='Checked Out' where id=%s"
        values = (id_,)
        return self.iud(qry, values)
```

```python
class Item(Connection):
    """
        This class works with menu for hotel.

        Methods:
            add_item()
            show_item()
            update_item()
            delete_item()
            search_item()

        """

    def __init__(self):
        super().__init__()

    def add_item(self, name, type_, price):
        """add an item to menu"""
        qry = "INSERT INTO items (name, type, price) VALUES (%s, %s, %s)"
        values = (name, type_, price)
        return self.iud(qry, values)

    def show_items(self):
        """returns all menu details"""
        qry = "SELECT * FROM items"
        res = self.show_data(qry)
        return res

    def update_item(self, id_, name, type_, price):
        """update an item detail of menu"""
        qry = "UPDATE items SET name = %s, type=%s, price =%s WHERE id = %s"
        values = (name, type_, price, id_)
        return self.iud(qry, values)

    def delete_item(self, id_):
        """delete an item from menu"""
        qry = "DELETE FROM items WHERE id = %s"
        values = (id_,)
        return self.iud(qry, values)

    def search_item(self, search):
        """search in menu"""
        qry = "SELECT * FROM items WHERE name LIKE %s OR type LIKE %s OR price LIKE %s"
        values = ("%" + search + "%", "%" + search + "%", "%" + search + "%")
        res = self.search_data(qry, values)
        return res
```

```python
class Order(Connection):
    """
        This class works with order of customer in hotel.

        Methods:
            add_order()
            show_order()
            delete_order()

        """

    def __init__(self):
        super().__init__()

    def add_order(self, ordered_item_list, cus_id):
        """save orders"""
        for i in ordered_item_list:
            qry = "INSERT INTO orders (item_id, cus_id, qty) VALUES (%s, %s, %s)"
            val = (i[0], cus_id, i[1])
            self.iud(qry, val)

    def show_orders(self, cus_id):
        """displays all orders made by a customer"""
        qry = """SELECT orders.id, items.type, items.name, orders.qty FROM orders
        JOIN items ON orders.item_id = items.id JOIN booking ON booking.id = orders.cus_id
        where cus_id = %s order by id desc"""
        values = (cus_id,)
        res = self.search_data(qry, values)
        return res

    def delete_order(self, order_id):
        """delete a specific order"""
        qry = "DELETE FROM orders WHERE id = %s"
        values = (order_id,)
        return self.iud(qry, values)
```

```
class Bill(Connection):
    """
            This class works with Billing of hotel.

            Methods:
                save_payment()
                room_bill()
                order_bill()
                total_order()
                view_bill()

            """

    def __init__(self):
        super().__init__()

    def save_payment(self, booking_id, discount, amount, paid, billed_by):
        """save billing details"""
        qry = "insert into payment (booking_id, discount, paid_amount, payment_type, billed_by)" \
              " values (%s, %s, %s, %s, %s)"
        values = (booking_id, discount, amount, paid, billed_by)
        return self.iud(qry, values)

    def room_bill(self, booking_id):
        """returns the customer details of stay in hotel and room for billing"""
        qry = "select cus_name, check_in, check_out, room_no, price " \
              "from booking join rooms on booking.room=rooms.room_no and booking.id=%s"
        values = (booking_id,)
        return self.search_data(qry, values)

    def order_bill(self, cus_id):
        """returns all the orders made by customer"""
        qry = "select concat(type,' ',name), price, qty, price*qty " \
              "from items join orders on orders.item_id = items.id where cus_id = %s;"
        values = (cus_id,)
        return self.search_data(qry, values)

    def total_order(self, cus_id):
        """returns the total orders bill"""
        qry = "select sum(price*qty) from items join orders on orders.item_id = items.id where cus_id = %s;"
        values = (cus_id,)
        return self.search_data(qry, values)

    def view_bill(self):
        """returns all the paid bills"""
        qry = """select cus_name, cus_add, cus_mobile, paid_amount, discount, payment_type, check_in, check_out,
         room, adults, children from payment join booking on payment.booking_id = booking.id"""
        value = self.show_data(qry)
        return value
```

DATABASE


Activities
MySQL Workbench
Sep 4 5:18 PM
MySQL Workbench
Local instance 3306
File Edit View Query Database Server Tools Scripting Help
Query 1
SQL File 4*
SQL File 5*
SQL File 6*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 10
Limit to 1000 rows
122
123 • select sum(price*qty) from items
124 join orders on orders.item_id = items.id where cus_id = 2;
125
126 • select cus_name, cus_add, cus_mobile, paid_amount, discount, payment_type, check_in, check_out, room, adults, children
127 from payment join booking on payment.booking_id = booking.id;
128
129 • show tables;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Tables_in_hotel
1 booking
2 items
3 orders
4 payment
5 rooms
6 staff
Result 1
Read Only
Query Completed

CONCLUSION

This application can help the small to medium business hotel and going digitalization protects their data and help the business to grow further. This Hotel Management System is easy and flexible to use. The error message lets any new user to operate the app smoothly. This app also has potential to develop further according to any hotels need which makes simpler by use of object oriented programming concept and use of database server.

REFERENCES

- Cormen, T. and Leiserson, C., n.d. Introduction To Algorithms, 3Rd Edition.
- Skiena, S., n.d. The Algorithm Design Manual.
- Tkdocs.com. 2020. Tkdocs - Tk Tutorial - Tree. [online] Available at: <https://tkdocs.com/tutorial/tree.html> [Accessed 4 September 2020].
- Python-textbok.readthedocs.io. 2020. Object-Oriented Programming In Python — Object-Oriented Programming In Python 1 Documentation. [online] Available at: <https://python-textbok.readthedocs.io/en/1.0/> [Accessed 4 September 2020].

10173127_Diptan_Regmi_22COM

ORIGINALITY REPORT

2%	0%	0%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Tresham College of Further and Higher Education Student Paper	2%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		