



COMPUTER REPORT

Hotel Management

NAME: Abhinav Singh

CLASS: XII

SCHOOL: Geethanjali Vidyalaya

ACADEMIC YEAR: 2022-23

CERTIFICATE GEETHANJALI VIDYALAYA



*This is to certify that the following is a bonafide
record of the computer project work done by
Aanand Sanjenbam of Geethanjali Vidyalaya,
under the guidance of Mrs. Seena
(Subject Teacher), Board of Secondary
Education (CBSE) during the academic year
2022-23*

EXTERNAL EXAMINER

INTERNAL EXAMINER

INSTITUTION STAMP

ACKNOWLEDGEMENT

This project would not have been possible without proper guidance and support of Mrs. Seena, my Computer teacher. I am highly indebted to her for sharing her indispensable time and knowledge to make this project a success.

I would also like to express my gratitude towards the school management and The Principal for providing me with best of the facilities and environment and giving such wonderful opportunities to shine.

Finally, I want to thank my classmates who have willingly helped me in every possible manner.

INDEX

S No.	CONTENTS	Page No.
1.	INTRODUCTION	5
2.	SYSTEM REQUIREMENTS	7
3.	SOURCE CODE	8
4.	OUTPUT	12
5.	REFERNCES	15

INTRODUCTION

Title of the project:

Hotel Management System

Prologue:

The main objective of this project is to provide the user an interface to book a hotel reservation. The program deals with storing the data of all hotel rooms.

The purpose of this project is the create an application program interface for management to access and update attributes of hotel rooms

The program also stores the details of any guests occupying a room in a database table.

Team Members:

- Aanand Sanjenbam
- Abhinav Singh
- Mann

Team details:

- The project “**Hotel Management System**” has been designed and developed by the 3 mentioned individuals. The various components of the project were shared between the members.

Limitations:

- None

SYSTEM

REQUIREMENTS

Recommended system requirements:

- Processor: Intel Core i3
- Disk Space: 2 to 4 GB
- OS: Windows 10
- Python version: 3.x.x
- MYSQL Command line client: 5.5 or higher

Minimum system requirements:

- Processor: Intel Atom or Intel i3
- Disk Space: 1 GB
- OS: Windows 7
- Python version: 2.7.x

Prerequisites before running the code:

- Python must be installed and, in the system's PATH
- MYSQL command line client is installed

SOURCE CODE

1. main.py

```
Main.py 2 Main.py~...
|
|  ##Importing Modules##
|
|  from faulthandler import disable
|  from sys import byteorder
|  from tkinter import *
|  import tkinter.font as font
|  from PIL import Image, ImageTk
|  import database as db
|  from Constants import *
|  from tkinter import messagebox
|
|  ##Success Output##
|
|  def messageboxy(fFloor=None):
|      messagebox.showinfo(title = "Title", message="Success!")
|      floorscreen(fFloor)
|
|  ##Room Information GUI##
|
|  def init_info_widgets(room_num, fFloor, status):
|      icon=booking_icon
|      lblfloor1= Label(info_frame, text=f"Room Number: {room_num}" , font="Helvetica 15 bold", bg=COLOUR_A, fg=COLOUR_B)
|      lblfloor1.place(x=10, y=200)
|      lblfloor2= Label(info_frame, text=f"Room Number: {fFloor}" , font="Helvetica 15 bold", image=display1, bg=INFO_FRAME_COLOUR)
|      lblfloor2.place(x=10, y=500)
|      Btn=Button(info_frame, font=myFont2, image=icon, borderwidth=0, height=100, width=150, text="Enter Booking \nDetails", compound
|
|      ##Fetching Data from Database##
|
|  def get_room_data(btn_text):
|      data = db.fetchdata(int(btn_text.strip()))[0]
|      init_info_widgets(data[0], data[1], data[2])
|
|      ##Room Number Creator##
|
|  def place_room_icons(fFloor_no, row, col):
|      i = 1
|      for y in range(row):
|          for x in range(col):
|              btn_text = f"{fFloor_no+i}"
|              if db.check_occ(int(btn_text)):
|                  icon=occupied_icon
|              else:
|                  icon=vacant_icon
|              Button(option_frame, font=myFont, image=icon, borderwidth=0, height=100, width=150, text=btn_text, compound="center",
|                  i+=1
|
|      ##Main GUI Interface##
|
|  def floorscreen(fFloor):
|      floor_no = fFloor*100
|      row, col = 3, 4
```



```

Main.py 2 Main.py\...

place_room_icons(floor_no, row, col)

lblfloor1= Label(option_frame, text=f"Floor {fFloor}", font="Helvetica 14 bold", bg=MID_FRAME_COLOUR)
lblfloor1.place(relx=0.46, y=630)
mainlbl1= Label(option_frame, text=f"ROOMS ON FLOOR {fFloor}", font="Helvetica 14 bold", bg=MID_FRAME_COLOUR, fg=SIDEBAR_COLOUR)
mainlbl1.place(x=50, y=10)

##Toggle for Floor Switching##

def arrowchecker():
    global floor
    arrowbtnright.place(relx= 0.7, y=625)
    arrowbtnleft.place(relx= 0.56, y=625)
    if floor==1:
        arrowbtnleft['state']= DISABLED
    elif floor==floors:
        arrowbtnright['state']= DISABLED
    else:
        arrowbtnright['state']= ACTIVE
        arrowbtnleft['state']= ACTIVE

    ##Initiation of Floor Switching##

def floorup():
    global floor
    floor+=1
    floorscreen(floor)
    arrowchecker()

def floordown():
    global floor
    floor-=1
    floorscreen(floor)
    arrowchecker()

##Constants##

iconsize = (150, 120)
arrowsize= (40,35)
floor=1
floors=4

##GUI Initiation##

win= Tk()
win.geometry("1000x700+500+150")

##Font Constants##

myFont = font.Font(size=20, family="Adobe Heiti Std R", weight="bold")
myFont2 = font.Font(size=10, family="Adobe Heiti Std R", weight="bold")

```

Main.py 2 Main.py\floordown

```
##GUI Building using Tkinter##

info_frame= Frame(win, bg=SIDEBAR_COLOUR)
info_frame.place(x=0, y=0, relwidth=0.3, relheight=1)
option_frame= Frame(bg=OPTION_FRAME_COLOUR)
option_frame.place(relx=0.3, rely= 0, relheight=1, relwidth=0.7)
colorframe1= Frame(option_frame, bg=MID_FRAME_COLOUR)
colorframe1.place(height=50, relwidth=1)
colorframe2= Frame(option_frame, bg= MID_FRAME_COLOUR)
colorframe2.place(y= 155, height=95, relwidth=1)
colorframe3= Frame(option_frame, bg= MID_FRAME_COLOUR)
colorframe3.place(y= 355, height=95, relwidth=1)
colorframe4= Frame(option_frame, bg= MID_FRAME_COLOUR)
colorframe4.place(y= 555, height=200, relwidth=1)

##Importing of program assets##

vacant_icon= PhotoImage(file= r"UI Elements\vacant_icon.png")
occupied_icon= PhotoImage(file= r'UI Elements\occupied_icon.png')
booking_icon= PhotoImage(file= r'UI Elements\booking.png')
right_icon=PhotoImage(file=r"UI Elements\right_arrow.png")
left_icon=PhotoImage(file=r"UI Elements\left_arrow.png")
display1=PhotoImage(file=r"UI Elements\Display1.png")

arrowbtnleft= Button(height= 30, width=50, image= left_icon, borderwidth=0, command= Lambda: floordown(), bg=MID_FRAME_COLOUR)
arrowbtnright= Button(height= 30, width=50, image= right_icon, borderwidth=0, command= Lambda: floorup(), bg= MID_FRAME_COLOUR)

floorscreen(1)
arrowchecker()

win.mainloop()
```

2. constants.py

```
Constants.py Constants.py\...
from PIL import Image, ImageTk
from tkinter import *

SIDEBAR_COLOUR="#394358"
OPTION_FRAME_COLOUR="#ecccfd"
MID_FRAME_COLOUR="#ebe5ca"
COLOUR_A="#262626"
COLOUR_B="#d6d3d1"

iconsize = (150, 120)
```

3. db.py

```
database.py database.py\ check_occ
import mysql.connector as sqltor
import random as r

##      Creating table      ##
mycon=sqltor.connect(host="localhost", user="root", passwd="Password@123", database="hotelmgmt")
cursor=mycon.cursor()
cursor.execute("CREATE TABLE if not exists attribute_table(hotel_num int primary key, floor int, room_type varchar(250))")
cursor.execute("CREATE TABLE if not exists transaction_table(hotel_num int primary key, occupancy_status int)")

##      Fetching Room Data      ##

def fetchdata(btn_num):
    cursor.execute(f"select * from attribute_table where hotel_num = {btn_num}")
    data=cursor.fetchall()
    return data

def check_occ(btn_num):
    cursor.execute(f"select occupancy_status from transaction_table where hotel_num={btn_num}")
    occupied=cursor.fetchall()[0][0]
    return int(occupied)

def updatestatus(room_num):
    occupied= check_occ(room_num)
    if occupied == 0:
        cursor.execute(f"update transaction_table set occupancy_status = 1 where hotel_num={room_num}")
    else:
        cursor.execute(f"update transaction_table set occupancy_status = 0 where hotel_num={room_num}")

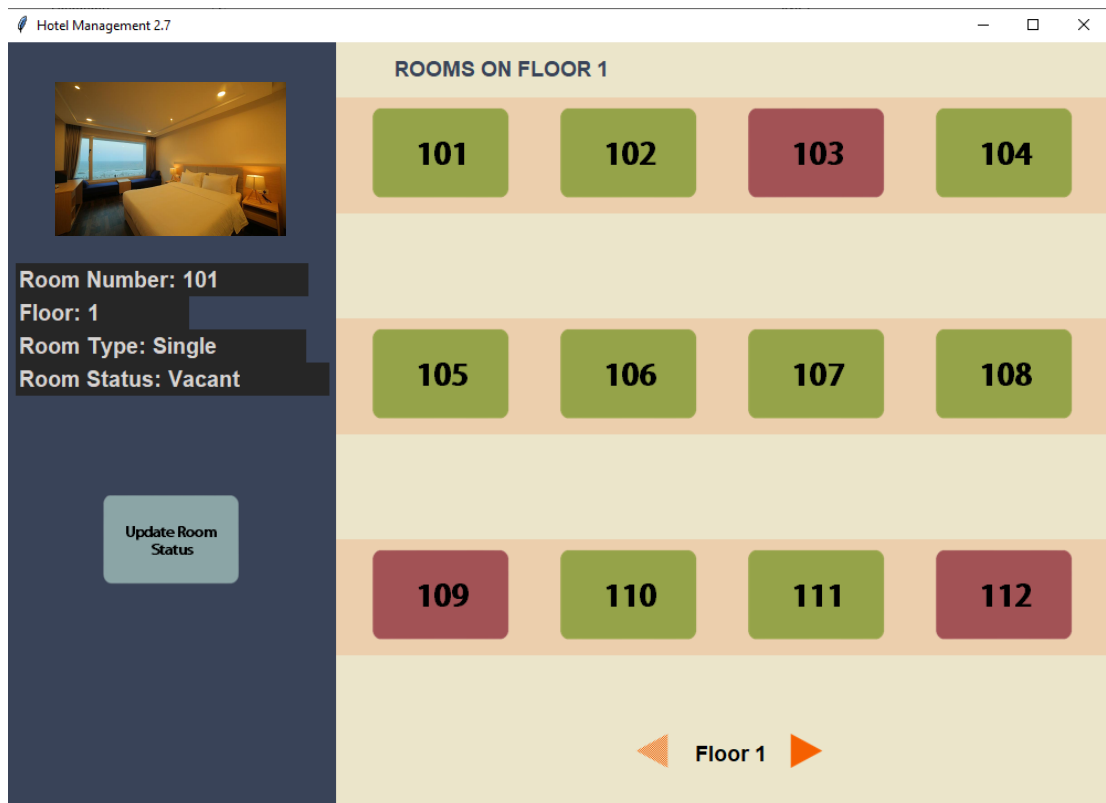
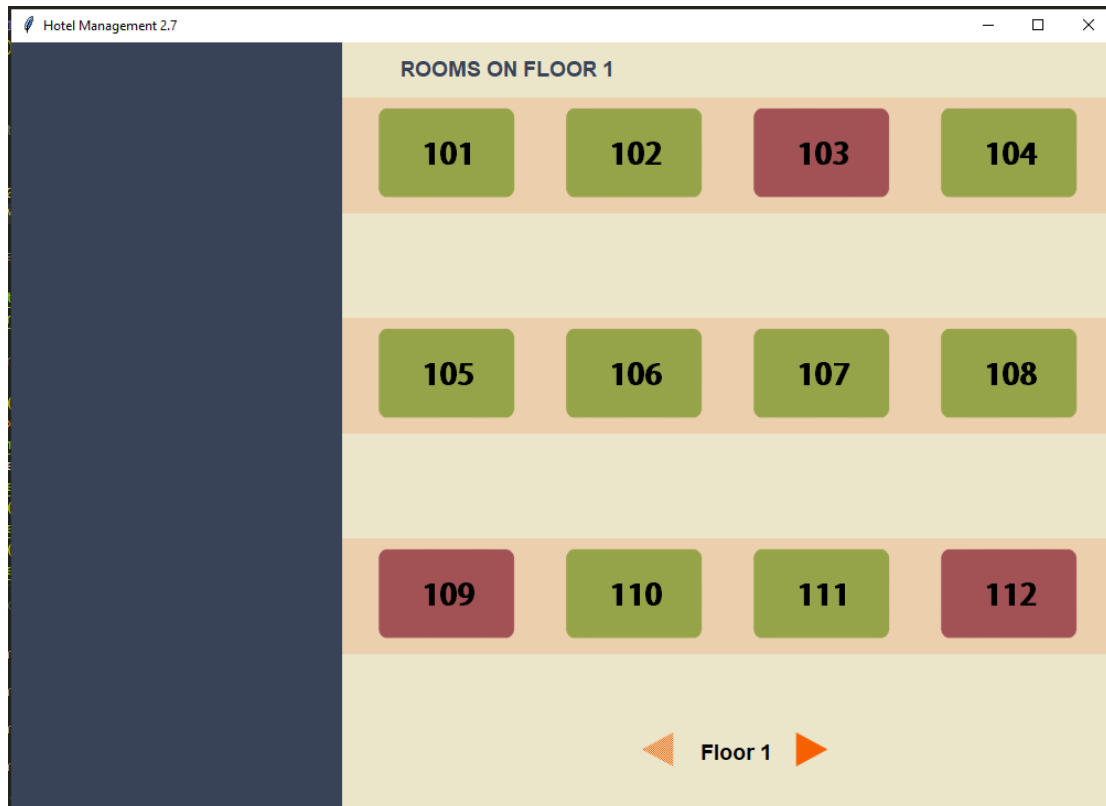
    mycon.commit()

##      Initializing Room Data      ##

if __name__=="__main__":
    try:
        for j in range(1,5):
            for i in range(j*100+1, j*100+13):
                cursor.execute(f"insert into transaction_table values({i}, {r.choice((0, 1))})")
                mycon.commit()
    except sqltor.errors.IntegrityError:
        pass

    try:
        for j in range(1,5):
            for i in range(j*100+1, j*100+13):
                if j*100+1<=i<=j*100+4:
                    type="Single"
                elif j*100+5<=i<=j*100+8:
                    type="Double"
                elif j*100+9<=i<=j*100+12:
                    type="Suite"
                cursor.execute(f"insert into attribute_table values({i}, {int(str(i)[0])}, '{type}')")
                mycon.commit()
    except sqltor.errors.IntegrityError:
        pass
```

OUTPUT



Hotel Management 2.7

ROOMS ON FLOOR 1

Room Number: 101
Floor: 1
Room Type: Single
Room Status: Vacant

Update Room Status

Acknowledgment
Room Status has been Updated!
OK

101 102 103 104

107 108

109 110 111 112

Floor 1

The screenshot shows the 'Hotel Management 2.7' application. On the left, a sidebar displays a room image and details for Room 101: Room Number: 101, Floor: 1, Room Type: Single, and Room Status: Vacant. Below this is an 'Update Room Status' button. The main area shows a grid of rooms on Floor 1, numbered 101 to 112. Rooms 101, 103, 109, and 112 are highlighted in red, indicating they are occupied. Rooms 102, 104, 107, 108, 110, and 111 are highlighted in green, indicating they are vacant. A confirmation dialog box is open in the center, stating 'Acknowledgment Room Status has been Updated!' with an 'OK' button. At the bottom of the grid, there are navigation arrows and the text 'Floor 1'.

Hotel Management 2.7

ROOMS ON FLOOR 1

Room Number: 101
Floor: 1
Room Type: Single
Room Status: Occupied

Update Room Status

101 102 103 104


105 106 107 108

109 110 111 112

Floor 1

This screenshot shows the same 'Hotel Management 2.7' application after the room status update. The sidebar now shows 'Room Status: Occupied' for Room 101. The main grid shows the same room layout, but the status of the rooms has changed: Rooms 101, 103, 109, and 112 are now red (occupied), while Rooms 102, 104, 105, 106, 107, 108, 110, and 111 are green (vacant). The confirmation dialog is no longer present. The 'Update Room Status' button remains in the sidebar. The 'Floor 1' navigation controls are at the bottom.

Hotel Management 2.7



Room Number: 206

Floor: 2

Room Type: Double

Room Status: Vacant


Update Room Status

ROOMS ON FLOOR 2

201	202	203	204
205	206	207	208
209	210	211	212

◀ Floor 2 ▶

Hotel Management 2.7



Room Number: 411

Floor: 4

Room Type: Suite

Room Status: Occupied

Update Room Status

ROOMS ON FLOOR 4

401	402	403	404
405	406	407	408
409	410	411	412

◀ Floor 4 ▶

Data Design

1. Sample Output

Query 1

1 select * from attribute_table

Limit to 1000 rows

Result Grid

	hotel_num	floor	room_type
	412	4	Suite
	411	4	Suite
	410	4	Suite
	409	4	Suite
	408	4	Double
	407	4	Double
	406	4	Double
	405	4	Double
	404	4	Single
	403	4	Single
	402	4	Single
	401	4	Single
	312	3	Suite
	311	3	Suite
	310	3	Suite
	309	3	Suite
	308	3	Double
	307	3	Double
	306	3	Double
	305	3	Double
	304	3	Single
	303	3	Single
	302	3	Single
	301	3	Single

2. Table structure

Query 1

1 select * from attribute_table

Limit to 1000 rows

Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision
1	hotel_num	hotelmgmt	attribute_table	INT	binary	11	
2	floor	hotelmgmt	attribute_table	INT	binary	11	
3	room_type	hotelmgmt	attribute_table	VARCHAR	utf8mb4	250	

REFERENCES

- Google
- Computer science with Sumita Aurora