

# **VITYARTHI PROJECT: HOTEL** **MANAGEMENT SYSTEM**

By  
**PRISHA ARORA**  
(25BAI10290)



**VIT Bhopal University**  
**Kothrikalan -466114,**  
**Madhya Pradesh,**  
**India**

### Algorithm:

1. Import Libraries and Modules: Import necessary Python libraries such as Tkinter for GUI, PIL for image handling, and custom modules for customer, room, and details functionalities.
2. Define Main Class that initializes the main window using Tkinter.
3. Initialize GUI Window to set the window's title and geometry (size).
4. Load and display several images (such as header, logo, menu background) using PIL and ImageTk.
5. Create a main frame to organize the GUI layout. Add a title label for the application.  
Create Menu and Buttons and add a sidebar/menu section with labeled buttons for different operations such as customer: opens customer details window; room: opens room booking window; details: opens room details window; logout: closes the application. Place each button in a dedicated frame.
6. Display Images and Layout: It used to load images for various visual sections of the application (top banner, sidebar, main area).
7. Define Button Commands: connect each menu button to a method that launches the corresponding functionality window using Toplevel windows.
8. Run Main Loop
9. Instantiate the hotel management system class.
10. Start the main event loop for the application.

## Main Program:

```

10 # main program (Type 3: bottom management system) 3_Jan_
11 from tkinter import *
12 from PIL import Image, ImageTk
13 from custom import cust_gui
14 from frame import frame_booking
15 from details import details_reservation
16 class bottommanagementSystem:
17     def __init__(self, root):
18         self.root=root
19         self.root.title("Hotel Management System")
20         self.root.geometry("1200x1000")
21
22 # ***** 1st frame *****
23 self.img=Image.open("C:\\Users\\bharati\\Desktop\\hotel_images\\imgset1.jpg")
24 self.img_resized=img.resize((500, 500), Image.LANCZOS)
25 self.photoimage=self.img.resize(self.photoimage.size, resample=Image.LANCZOS)
26 self.img_label(self.root, image=self.photoimage, hd=4, relief=RIDGE)
27 self.l1.place(x=0,y=0,width=500,height=500)
28 # ***** 2nd frame *****
29 self.img=Image.open("C:\\Users\\bharati\\Desktop\\hotel_images\\imgset2.jpg")
30 self.img_resized=img.resize((500, 500), Image.LANCZOS)
31 self.photoimage=self.img.resize(self.photoimage.size, resample=Image.LANCZOS)
32 self.img_label(self.root, image=self.photoimage, hd=4, relief=RIDGE)
33 self.l2.place(x=0,y=0,width=500,height=500)
34 # ***** 3rd frame *****
35 self.l1.titleLabel(self.root, text="HOTEL MANAGEMENT SYSTEM", font="Times new roman", 40, "bold", bg="black", fg="gold", hd=4, relief=RIDGE)
36 self.l1.title.place(x=0,y=150,width=1200,height=60)
37 # ***** 4th frame *****
38 self.main_frame=Frame(self.root, hd=4, relief=RIDGE)
39 self.main_frame.place(x=0,y=150,width=1200,height=950)
40 # ***** 5th frame *****
41 self.menu_label(self.main_frame, text="MENU", font="Times new roman", 20, "bold", bg="black", fg="gold", hd=4, relief=RIDGE)
42 self.l3.place(x=0,y=0,width=300)
43 # ***** 6th frame *****
44 self.frame_booking=Frame(self.main_frame, hd=4, relief=RIDGE)
45 self.frame_booking.place(x=0,y=40,width=720,height=360)
46
47 # ***** 7th frame *****
48 cust_gui.booking_details_frame, hotel="SYSTEM", custom=self.cust_details,width=72,height="Times new roman", 17, "bold", bg="black", fg="gold", hd=4, cursor="hand2")

```

```

37 cur = tk.Canvas(master, frame, width=column0, command=self.cur_details,width=17,height=100,highlightthickness=1,highlightcolor="black",
38               cursor="hand2",
39               bg="white",relief="solid",
40               borderwidth=1,font=("times new roman",12,"bold"),bg="black",fg="pink",bd=0,cursor="hand2")
41
42 row=tk.Button(master,frame, text="row",command=self.rowbooking,width=17,height=100,highlightthickness=1,highlightcolor="black",
43               cursor="hand2",
44               bg="white",relief="solid",
45               borderwidth=1,font=("times new roman",12,"bold"),bg="black",fg="pink",bd=0,cursor="hand2")
46
47 details=tk.Button(master,frame, text="details",command=self.details_row,width=17,height=100,highlightthickness=1,highlightcolor="black",
48                  cursor="hand2",
49                  bg="white",relief="solid",
50                  borderwidth=1,font=("times new roman",12,"bold"),bg="black",fg="pink",bd=0,cursor="hand2")
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
10
```

```

67 lblimg=Label(master_frame,image=self.photoimg,bd=4,relief=RIDGE)
68 lblimg.place(x=0,y=0,width=150,height=100)
69
70 def cust_details(self):
71     self.new_window=tk.Toplevel(self.root)
72     self.app=ttk.Frame(self.new_window)
73
74     def roombooking(self):
75         self.new_window=tk.Toplevel(self.root)
76         self.app=roombooking(self.new_window)
77
78     def details_room(self):
79         self.new_window=tk.Toplevel(self.root)
80         self.app=detailsroom(self.new_window)
81
82     def login(self):
83         self.root.destroy()
84
85 if __name__ == '__main__':
86     root=tk()
87     objHotelManagementSystem(root)
88     root.mainloop()

```

## Customer Details Program:

```

1 from tkinter import*
2 from PIL import Image,ImageTk
3 from tkinter import ttk
4 import random
5 import mysql.connector
6 from tkinter import messagebox
7
8 class cust_win:
9     def __init__(self,root):
10         self.root=root
11         self.root.title("Hotel Management System")
12         self.root.geometry("1600x700+230+105")
13         #=====var labels=====
14         self.var_ref=StringVar()
15         x=random.randint(1000,9999)
16         self.var_ref.set(str(x))
17
18         self.var_cust_name=StringVar()
19         self.var_mother=StringVar()
20         self.var_gender=StringVar()
21         self.var_post=StringVar()
22         self.var_mobile=StringVar()
23         self.var_nationality=StringVar()
24         self.var_id_proof=StringVar()
25         self.var_id_number=StringVar()
26         self.var_address=StringVar()
27         #=====img=====
28         lbl_title=Label(self.root,text="ADD CUSTOMER DETAILS",font=("times new roman",30,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE)
29         lbl_title.place(x=0,y=0,width=1600,height=60)
30         #=====img=====
31         img1=Image.open("C:\Users\Akhin\Desktop\hotel images\logo.png")
32         img1=ImageTk.PhotoImage(img1)
33         self.photoimg=img1
34         lblimg=Label(self.root,image=self.photoimg,bd=4,relief=RIDGE)
35         lblimg.place(x=0,y=0,width=150,height=100)
36         #=====var labels=====
37         labelframeleft=tk.Frame(self.root,bd=2,relief=RIDGE,text="Customers Details",font=("times new roman",22,"bold"),pady=2)

```

```

38 labelframeleft.place(x=0,y=0,width=1600,height=600)
39 #=====var labels=====
40 self.var_ref=StringVar()
41 lbl_cust_ref=Label(labelframeleft,text="Customer Ref",font=("times new roman",20,"bold"),pady=2,padx=4)
42 lbl_cust_ref.grid(row=0,column=0,sticky=W)
43 entry_ref=tk.Entry(labelframeleft,width=24,textvariable=self.var_ref,font=("times new roman",21,"bold"),state="readonly")
44 entry_ref.grid(row=0,column=1)
45 #=====var labels=====
46 name=Label(labelframeleft,text="Customer Name",font=("times new roman",20,"bold"),pady=2,padx=4)
47 name.grid(row=1,column=0,sticky=W)
48 txtname=tk.Entry(labelframeleft,width=24,textvariable=self.var_cust_name,font=("times new roman",21,"bold"))
49 txtname.grid(row=1,column=1)
50 #=====var labels=====
51 hname=Label(labelframeleft,text="Mother Name",font=("times new roman",20,"bold"),pady=2,padx=4)
52 hname.grid(row=2,column=0,sticky=W)
53 txtname=tk.Entry(labelframeleft,width=24,textvariable=self.var_mother,font=("times new roman",21,"bold"))
54 txtname.grid(row=2,column=1)
55 #=====var labels=====
56 gender=Label(labelframeleft,text="Gender",font=("times new roman",20,"bold"),pady=2,padx=4)
57 gender.grid(row=3,column=0,sticky=W)
58 combo_gender=tk.Combobox(labelframeleft,width=23,textvariable=self.var_gender,font=("times new roman",21,"bold"),state="readonly")
59 combo_gender['values']=("Male","Female","Other")
60 combo_gender.current(0)
61 combo_gender.grid(row=3,column=1)
62 #=====var labels=====
63 hpostcode=Label(labelframeleft,text="Postcode",font=("times new roman",20,"bold"),pady=2,padx=4)
64 hpostcode.grid(row=4,column=0,sticky=W)
65 txtpostcode=tk.Entry(labelframeleft,width=24,textvariable=self.var_post,font=("times new roman",21,"bold"))
66 txtpostcode.grid(row=4,column=1)
67 #=====var labels=====
68 hmobile=Label(labelframeleft,text="Mobile",font=("times new roman",20,"bold"),pady=2,padx=4)
69 hmobile.grid(row=5,column=0,sticky=W)
70 txtmobile=tk.Entry(labelframeleft,width=24,textvariable=self.var_mobile,font=("times new roman",21,"bold"))
71 txtmobile.grid(row=5,column=1)
72 #=====var labels=====
73 hnationality=Label(labelframeleft,text="Nationality",font=("times new roman",20,"bold"),pady=2,padx=4)
74 hnationality.grid(row=6,column=0,sticky=W)
75 combo_nationality=tk.Combobox(labelframeleft,width=23,textvariable=self.var_nationality,font=("times new roman",21,"bold"),state="readonly")
76 combo_nationality['values']=("Indian","American","Russian")
77 combo_nationality.current(0)
78 combo_nationality.grid(row=6,column=1)
79 #=====var labels=====
80 hproof_type=Label(labelframeleft,text="ID Proof Type",font=("times new roman",20,"bold"),pady=2,padx=4)
81 hproof_type.grid(row=7,column=0,sticky=W)
82 combo_idproof=tk.Combobox(labelframeleft,width=23,textvariable=self.var_id_proof,font=("times new roman",21,"bold"),state="readonly")
83 combo_idproof['values']=("Aadhaarcard","Driving license","Passport")
84 combo_idproof.current(0)
85 combo_idproof.grid(row=7,column=1)
86 #=====var labels=====
87 hnumber=Label(labelframeleft,text="ID Number",font=("times new roman",20,"bold"),pady=2,padx=4)
88 hnumber.grid(row=8,column=0,sticky=W)
89 txtidnumber=tk.Entry(labelframeleft,width=24,textvariable=self.var_id_number,font=("times new roman",21,"bold"))
90 txtidnumber.grid(row=8,column=1)
91 #=====var labels=====
92 haddress=Label(labelframeleft,text="Address",font=("times new roman",22,"bold"),pady=2,padx=4)
93 haddress.grid(row=9,column=0,sticky=W)
94 txtaddress=tk.Entry(labelframeleft,width=24,textvariable=self.var_address,font=("times new roman",21,"bold"))
95 txtaddress.grid(row=9,column=1)

```

```

70 #=====var labels=====
71 hmobile=Label(labelframeleft,text="Mobile",font=("times new roman",20,"bold"),pady=2,padx=4)
72 hmobile.grid(row=5,column=0,sticky=W)
73 txtmobile=tk.Entry(labelframeleft,width=24,textvariable=self.var_mobile,font=("times new roman",21,"bold"))
74 txtmobile.grid(row=5,column=1)
75 #=====var labels=====
76 hnationality=Label(labelframeleft,text="Nationality",font=("times new roman",20,"bold"),pady=2,padx=4)
77 hnationality.grid(row=6,column=0,sticky=W)
78 combo_nationality=tk.Combobox(labelframeleft,width=23,textvariable=self.var_nationality,font=("times new roman",21,"bold"),state="readonly")
79 combo_nationality['values']=("Indian","American","Russian")
80 combo_nationality.current(0)
81 combo_nationality.grid(row=6,column=1)
82 #=====var labels=====
83 hproof_type=Label(labelframeleft,text="ID Proof Type",font=("times new roman",20,"bold"),pady=2,padx=4)
84 hproof_type.grid(row=7,column=0,sticky=W)
85 combo_idproof=tk.Combobox(labelframeleft,width=23,textvariable=self.var_id_proof,font=("times new roman",21,"bold"),state="readonly")
86 combo_idproof['values']=("Aadhaarcard","Driving license","Passport")
87 combo_idproof.current(0)
88 combo_idproof.grid(row=7,column=1)
89 #=====var labels=====
90 hnumber=Label(labelframeleft,text="ID Number",font=("times new roman",20,"bold"),pady=2,padx=4)
91 hnumber.grid(row=8,column=0,sticky=W)
92 txtidnumber=tk.Entry(labelframeleft,width=24,textvariable=self.var_id_number,font=("times new roman",21,"bold"))
93 txtidnumber.grid(row=8,column=1)
94 #=====var labels=====
95 haddress=Label(labelframeleft,text="Address",font=("times new roman",22,"bold"),pady=2,padx=4)
96 haddress.grid(row=9,column=0,sticky=W)
97 txtaddress=tk.Entry(labelframeleft,width=24,textvariable=self.var_address,font=("times new roman",21,"bold"))
98 txtaddress.grid(row=9,column=1)

```

```

101 #*****adding*****
102 btn_frame=Frame(labelFrame1,bd=1,relief=RIDGE)
103 btn_frame.place(x=0,y=470,width=650,height=50)
104
105 btnadd=Button(btn_frame,text='ADD',command=self.add_data,font=("times new roman",18,"bold"),bg='black',fg='gold',width=9)
106 btnadd.grid(row=0,column=0,padx=1)
107
108 btnupdate=Button(btn_frame,text='UPDATE',command=self.update,font=("times new roman",18,"bold"),bg='black',fg='gold',width=18)
109 btnupdate.grid(row=0,column=1,padx=1)
110
111 btndelete=Button(btn_frame,text='DELETE',command=self.delete,font=("times new roman",18,"bold"),bg='black',fg='gold',width=9)
112 btndelete.grid(row=0,column=2,padx=1)
113
114 btnreset=Button(btn_frame,text='RESET',command=self.reset,font=("times new roman",18,"bold"),bg='black',fg='gold',width=10)
115 btnreset.grid(row=0,column=3,padx=1)
116
117 #*****table frame search system*****
118 table_frame=LabelFrame(self.root,bd=2,relief=RIDGE,text='View Details and Search System',font=("times new roman",22,"bold"),padx=2)
119 table_frame.place(x=585,y=60,width=1110,height=600)
120
121 lblsearchby=Label(table_frame,text='Search By:',font=("times new roman",22,"bold"),bg='red',fg='white')
122 lblsearchby.grid(row=0,column=0,sticky=N,padx=1)
123 self.search_var=StringVar()
124 combo_search=ttk.Combobox(table_frame,width=19,textvariable=self.search_var,font=("times new roman",22,"bold"),state='readonly')
125 combo_search['value']=('name','mobile')
126 combo_search.current(0)
127 combo_search.grid(row=0,column=1,padx=1)
128
129 self.txt1_search=StringVar()
130 txtsearch=ttk.Entry(table_frame,width=19,textvariable=self.txt1_search,font=("times new roman",22,"bold"))
131 txtsearch.grid(row=0,column=2,padx=1)
132
133 btnsearch=Button(table_frame,text='Search',command=self.search,font=("times new roman",18,"bold"),bg='black',fg='gold',width=10)

```

```

134 btnsearch.grid(row=0,column=3,padx=1)
135
136 btnshowall=Button(table_frame,text='Show All',command=self.fetch_data,font=("times new roman",18,"bold"),bg='black',fg='gold',width=10)
137 btnshowall.grid(row=0,column=4,padx=1)
138
139 #*****show data*****
140 details=TableFrame(table_frame,bd=1,relief=RIDGE)
141 details.table.place(x=0,y=45,width=1100,height=600)
142
143 scroll_y=ttk.Scrollbar(details.table,width=40,orient=VERTICAL)
144 scroll_y.grid(row=0,column=0,sticky=N+S)
145 self.cust_details=Table(tk,scroll(details.table,columns=('ref','name','mother','gender','ref','mobile','nationality','idproof','idnumber','address'),scrollcommand=scroll_x.set,scrollcommand=scroll_y.set))
146 scroll_x.pack(side=HORIZONTAL,fill=X)
147 scroll_y.pack(side=VERTICAL,fill=Y)
148 scroll_x.config(command=self.cust_details.table.xview)
149 scroll_y.config(command=self.cust_details.table.yview)
150
151 self.cust_details.table.heading('ref',text='refer no')
152 self.cust_details.table.heading('name',text='name')
153 self.cust_details.table.heading('mother',text='mother name')
154 self.cust_details.table.heading('gender',text='gender')
155 self.cust_details.table.heading('post',text='post code')
156 self.cust_details.table.heading('mobile',text='mobile')
157 self.cust_details.table.heading('nationality',text='nationality')
158 self.cust_details.table.heading('idproof',text='id proof')
159 self.cust_details.table.heading('idnumber',text='id number')
160 self.cust_details.table.heading('address',text='address')
161
162 self.cust_details.table['show'].headings()
163
164 self.cust_details.table.column('ref',width=110)
165 self.cust_details.table.column('name',width=110)
166 self.cust_details.table.column('mother',width=110)
167 self.cust_details.table.column('gender',width=110)

```

```

168 self.cust_details.table.column('post',width=110)
169 self.cust_details.table.column('mobile',width=110)
170 self.cust_details.table.column('nationality',width=110)
171 self.cust_details.table.column('idproof',width=110)
172 self.cust_details.table.column('idnumber',width=110)
173 self.cust_details.table.column('address',width=110)
174 self.cust_details.table.pack(fill=BOTH,expand=1)
175 self.cust_details.table.bind('<ButtonRelease-1>',self.get_cursor)
176
177 #*****Add data*****
178 def add_data(self):
179     if self.var.mobile.get()==" or self.var.mother.get()=="":
180         messagebox.showerror("ERROR","All Fields are Required",parent=self.root)
181     else:
182         try:
183             conn=mysql.connector.connect(host='localhost',username='root',password='mornil',database='student1')
184             my_cursor=conn.cursor()
185             my_cursor.execute("insert into customer values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
186                               ,(self.var_ref.get(),
187                                self.var_name.get(),
188                                self.var_mother.get(),
189                                self.var_gender.get(),
190                                self.var_post.get(),
191                                self.var_mobile.get(),
192                                self.var_nationality.get(),
193                                self.var_id_proof.get(),
194                                self.var_id_number.get(),
195                                self.var_address.get()
196                               ))
197             conn.commit()
198             self.fetch_data()
199             conn.close()
200             messagebox.showinfo("Success","Customer has been added",parent=self.root)

```

```

202         except Exception as e:
203             messagebox.showwarning("Warning",f"Something went wrong:{str(e)}",parent=self.root)
204     #*****Fetch data*****
205     def fetch_data(self):
206         conn=mysql.connector.connect(host="localhost",username="host",password="morni",database="student1")
207         my_cursor=conn.cursor()
208         my_cursor.execute("select * from customer")
209         rows=my_cursor.fetchall()
210         if len(rows)!=0:
211             self.cust_details_table.delete(*self.cust_details_table.get_children())
212             for i in rows:
213                 self.cust_details_table.insert("",END,values=i)
214             conn.commit()
215             conn.close()
216     #*****Get cursor*****
217     def get_cursor(self,event=""):
218         cursor_row=self.cust_details_table.focus()
219         content=self.cust_details_table.item(cursor_row)
220         row=content["values"]
221
222         self.var_ref.set(row[0]),
223         self.var_cust_name.set(row[1]),
224         self.var_mother.set(row[2]),
225         self.var_gender.set(row[3]),
226         self.var_post.set(row[4]),
227         self.var_mobile.set(row[5]),
228         self.var_nationality.set(row[6]),
229         self.var_id_proof.set(row[7]),
230         self.var_id_number.set(row[8]),
231         self.var_address.set(row[9])
232     #*****Update*****
233     def update(self):
234         if self.var_mobile.get()=="":
235             messagebox.showerror("Error","Please enter mobile number",parent=self.root)

```

```

236     elif:
237         conn=mysql.connector.connect(host="localhost",username="host",password="morni",database="student1")
238         my_cursor=conn.cursor()
239         my_cursor.execute("update customer set name=%s,mother=%s,sender=%s,postcode=%s,rollno=%s,nationality=%s,idproof=%s,idnumber=%s,address=%s where ref=%s",[
240             self.var_ref.get(),
241             self.var_cust_name.get(),
242             self.var_mother.get(),
243             self.var_gender.get(),
244             self.var_post.get(),
245             self.var_mobile.get(),
246             self.var_nationality.get(),
247             self.var_id_proof.get(),
248             self.var_id_number.get(),
249             self.var_address.get(),
250             self.var_ref.get()
251         ])
252         conn.commit()
253         self.fetch_data()
254         conn.close()
255         messagebox.showinfo("Update","Customer details has been updated successfully",parent=self.root)
256     #*****Delete*****
257     def delete(self):
258         delete=messagebox.askyesno("WITH MANAGEMENT SYSTEM","Do you want delete this customer",parent=self.root)
259         if delete==yes:
260             conn=mysql.connector.connect(host="localhost",username="host",password="morni",database="student1")
261             my_cursor=conn.cursor()
262             query="delete from customer where ref=%s"
263             values(self.var_ref.get())
264             my_cursor.execute(query,values)
265         elif:
266             return
267         conn.commit()
268         self.fetch_data()
269         conn.close()

```

```

270     #*****Reset*****
271     def reset(self):
272         self.var_ref.set("")
273         self.var_cust_name.set("")
274         self.var_mother.set("")
275         self.var_gender.set("")
276         self.var_post.set("")
277         self.var_mobile.set("")
278         self.var_nationality.set("")
279         self.var_id_proof.set("")
280         self.var_id_number.set("")
281         self.var_address.set("")
282
283         x=random.randint(1000,9999)
284         self.var_ref.set(str(x))
285     #*****Search*****
286     def search(self):
287         conn=mysql.connector.connect(host="localhost",username="host",password="morni",database="student1")
288         my_cursor=conn.cursor()
289
290         my_cursor.execute("select * from customer where "+str(self.search_var.get())+" (LIKE '%"+str(self.txt1_search.get())+"%')")
291         rows=my_cursor.fetchall()
292         if len(rows)!=0:
293             self.cust_details_table.delete(*self.cust_details_table.get_children())
294             for i in rows:
295                 self.cust_details_table.insert("",END,values=i)
296             conn.commit()
297             conn.close()
298
299 if __name__=="__main__":
300     root=Tk()
301     obj=Cust_win(root)
302     root.mainloop()

```

## Room Program:

```

# example 3: Tkinter window 1 (GUI) .py
1 from tkinter import *
2 from tkinter import messagebox
3 from tkinter import ttk
4 import random
5 import mysql.connector
6 from tkinter import messagebox
7 from time import strftime
8 from datetime import datetime
9
10 class RoomBooking:
11     def __init__(self,root):
12         self.root=root
13         self.root.title("Hotel Management System")
14         self.root.geometry("1200x800")
15     #*****Get cursor*****
16     def var_contact(self,input):
17         self.var_contact=self.input.get()
18     def var_gender(self,input):
19         self.var_gender=self.input.get()
20     def var_post(self,input):
21         self.var_post=self.input.get()
22     def var_mobile(self,input):
23         self.var_mobile=self.input.get()
24     def var_nationality(self,input):
25         self.var_nationality=self.input.get()
26     def var_id_proof(self,input):
27         self.var_id_proof=self.input.get()
28     def var_id_number(self,input):
29         self.var_id_number=self.input.get()
30     def var_address(self,input):
31         self.var_address=self.input.get()
32     #*****Update*****
33     def update(self):
34         conn=mysql.connector.connect(host="localhost",username="host",password="morni",database="student1")
35         my_cursor=conn.cursor()
36         my_cursor.execute("update customer set name=%s,mother=%s,sender=%s,postcode=%s,rollno=%s,nationality=%s,idproof=%s,idnumber=%s,address=%s where ref=%s",[
37             self.var_ref.get(),
38             self.var_cust_name.get(),
39             self.var_mother.get(),
40             self.var_gender.get(),
41             self.var_post.get(),
42             self.var_mobile.get(),
43             self.var_nationality.get(),
44             self.var_id_proof.get(),
45             self.var_id_number.get(),
46             self.var_address.get(),
47             self.var_ref.get()
48         ])
49         conn.commit()
50         self.fetch_data()
51         conn.close()
52
53 if __name__=="__main__":
54     root=Tk()
55     obj=Cust_win(root)
56     root.mainloop()

```





```

10 # roombooking > @ _init_
11 class roombooking:
12     def __init__(self,root):
13
14         self.root=ttk.Treeview(details_table,column=('Contact','Checkin','Checkout','Roomtype','Roomavailable','Meal','No of days'),
15                                xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
16         scroll_x.pack(side=TOTTOM,fill=X)
17         scroll_y.pack(side=LEFT,fill=Y)
18         scroll_x.config(command=self.root.table.xview)
19         scroll_y.config(command=self.root.table.yview)
20
21         self.root.table.heading('Contact',text='Contact')
22         self.root.table.heading('Checkin',text='Checkin')
23         self.root.table.heading('Checkout',text='Checkout')
24         self.root.table.heading('Roomtype',text='Roomtype')
25         self.root.table.heading('Roomavailable',text='Room no')
26         self.root.table.heading('Meal',text='Meal')
27         self.root.table.heading('No of days',text='No of days')
28
29         self.root.table['show']='headings'
30
31         self.root.table.column('Contact',width=110)
32         self.root.table.column('Checkin',width=110)
33         self.root.table.column('Checkout',width=110)
34         self.root.table.column('Roomtype',width=110)
35         self.root.table.column('Roomavailable',width=110)
36         self.root.table.column('Meal',width=110)
37         self.root.table.column('No of days',width=110)
38         self.root.table.pack(fill=BOTH,expand=1)
39         self.root.table.bind('<ButtonRelease-1>',self.get_cursor)
40         self.fetch_data()
41
42         #=====add data function=====
43         def add_data(self):
44             if self.var_contact.get()==" " or self.var_checkin.get()==" ":
45                 messagebox.showerror("ERROR","All fields are required",parent=self.root)
46             else:
47                 try:
48                     conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
49
50

```

```

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

10 # roombooking > @ _init_
11 class roombooking:
12     def __init__(self,root):
13
14         self.var_contact.set(row[0])
15         self.var_checkin.set(row[1])
16         self.var_checkout.set(row[2])
17         self.var_roomtype.set(row[3])
18         self.var_roomavailable.set(row[4])
19         self.var_meal.set(row[5])
20         self.var_nofdays.set(row[6])
21
22         #=====update function=====
23         def update(self):
24             if self.var_contact.get()==" ":
25                 messagebox.showerror("Error","Please enter contact number",parent=self.root)
26             else:
27                 conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
28                 my_cursor=conn.cursor()
29                 my_cursor.execute("update room set check_in=%d,check_out=%d,roomtype=%d,roomavailable=%d,meal=%d,noofdays=%d where Contact=%d",(
30                                     self.var_checkin.get(),
31                                     self.var_checkout.get(),
32                                     self.var_roomtype.get(),
33                                     self.var_roomavailable.get(),
34                                     self.var_meal.get(),
35                                     self.var_nofdays.get(),
36                                     self.var_contact.get()
37                                     ))
38                 conn.commit()
39                 self.fetch_data()
40                 conn.close()
41                 messagebox.showinfo("update","Room details has been updated successfully",parent=self.root)
42
43         #=====delete function=====
44         def delete(self):
45             delete=messagebox.askyesno("HOTEL MANAGEMENT SYSTEM","Do you want delete this customer",parent=self.root)
46             if delete==True:
47                 conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
48                 my_cursor=conn.cursor()
49                 query="delete from room where Contact=%d"
50
51

```

```

10 # roombooking > @ _init_
11 class roombooking:
12     def __init__(self,root):
13
14         self.var_contact.set(row[0])
15         self.var_checkin.set(row[1])
16         self.var_checkout.set(row[2])
17         self.var_roomtype.set(row[3])
18         self.var_roomavailable.set(row[4])
19         self.var_meal.set(row[5])
20         self.var_nofdays.set(row[6])
21
22         #=====delete function=====
23         def delete(self):
24             delete=messagebox.askyesno("HOTEL MANAGEMENT SYSTEM","Do you want delete this customer",parent=self.root)
25             if delete==True:
26                 conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
27                 my_cursor=conn.cursor()
28                 query="delete from room where Contact=%d"
29                 my_cursor.execute(query)
30                 conn.commit()
31                 self.fetch_data()
32                 conn.close()
33                 messagebox.showinfo("delete","This customer has been deleted",parent=self.root)
34             else:
35                 conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
36                 my_cursor=conn.cursor()
37                 query="select * from room where Contact=%d"
38                 my_cursor.execute(query)
39                 rows=my_cursor.fetchall()
40
41                 if len(rows)>0:
42                     conn=mysql.connector.connect(host="localhost",username="root",password="yash5",database="management")
43                     my_cursor=conn.cursor()
44                     query="delete from room where Contact=%d"
45                     my_cursor.execute(query)
46                     conn.commit()
47                     self.fetch_data()
48                     conn.close()
49                     messagebox.showinfo("delete","This customer has been deleted",parent=self.root)
50                 else:
51                     messagebox.showerror("Error","This customer does not exist",parent=self.root)
52

```

```

5  # Import the necessary modules
6  import sys
7  import random
8  import math
9
10 # Define the main function
11 def main():
12     # Create a list of numbers
13     numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
14
15     # Shuffle the list
16     random.shuffle(numbers)
17
18     # Print the shuffled list
19     print("Shuffled list:", numbers)
20
21     # Calculate the sum of the numbers
22     total = 0
23     for number in numbers:
24         total += number
25
26     # Print the sum
27     print("Sum of numbers:", total)
28
29 # Call the main function
30 if __name__ == "__main__":
31     main()

```



```

10: www.pythontesting.com - 10_04_
11: class roombooking:
12:     def __init__(self):
13:
14:         self.var_mail_get[0]="bookit" and self.var_roomtype.get[0]="single":
15:             qd=load(500)
16:             qd=load(500)
17:             qd=load(self.var_roomtypes.get[0])
18:             qd=load(qd.get)
19:             qd=load(qd.get)
20:             fac="N", "set(1,1,12, (qd["N"],1))
21:             str="N", "set(1,1,12, (qd["N"],1))
22:             str="N", "set(1,1,12, (qd["N"],1))
23:             self.var_mail_get.set(1)
24:             self.var_actualtotal.set(0)
25:             self.var_total.set(1)
26:
27:         self.var_mail_get[0]="bookroom" and self.var_roomtype.get[0]="single":
28:             qd=load(500)
29:             qd=load(500)
30:             qd=load(self.var_roomtypes.get[0])
31:             qd=load(qd.get)
32:             qd=load(qd.get)
33:             fac="N", "set(1,1,12, (qd["N"],1))
34:             str="N", "set(1,1,12, (qd["N"],1))
35:             str="N", "set(1,1,12, (qd["N"],1))
36:             self.var_mail_get.set(1)
37:             self.var_actualtotal.set(0)
38:             self.var_total.set(1)
39:
40:         self.var_mail_get[0]="bookflat" and self.var_roomtype.get[0]="bookit":
41:             qd=load(500)
42:             qd=load(500)
43:             qd=load(self.var_roomtypes.get[0])
44:             qd=load(qd.get)
45:             qd=load(qd.get)
46:             fac="N", "set(1,1,12, (qd["N"],1))
47:             str="N", "set(1,1,12, (qd["N"],1))
48:             str="N", "set(1,1,12, (qd["N"],1))

```

[illegible][illegible][illegible]

### Details:

```

2 from tkinter import *
3 from PIL import Image, ImageTk
4 from tkinter import ttk
5 import random
6 import mysql.connector
7 from tkinter import messagebox
8 from time import strftime
9 from functools import partial
10
11 class DetailRoom:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Hotel Management System")
15         self.root.geometry("1000x1000")
16
17         # ***** Room Details *****
18         lbl_title = Label(self.root, text="ROOM DETAILS", font=("times new roman", 22, "bold"), bg="black", fg="gold", bd=1, relief="solid")
19         lbl_title.place(x=0, y=0, width=1000, height=50)
20
21         # ***** Room Image *****
22         img_image = ImageTk.PhotoImage(Image.open("hotel.jpg"))
23         img_label = Label(self.root, image=img_image, width=1000, height=100)
24         img_label.place(x=0, y=50, width=1000, height=100)
25
26         # ***** Room Info *****
27         lbl_room_info = Label(self.root, text="Room Info", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2)
28         lbl_room_info.place(x=0, y=100, width=1000, height=100)
29
30         # ***** Room Type *****
31         lbl_room_type = Label(self.root, text="Room Type", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2, pady=2)
32         lbl_room_type.grid(row=0, column=0, sticky="w")
33
34         self.var_floor = StringVar()
35         entry_floor = ttk.Entry(lbl_room_type, width=17, textvariable=self.var_floor, font=("times new roman", 19, "bold"))
36         entry_floor.grid(row=0, column=1, sticky="w")
37
38         # ***** Room No *****
39         lbl_room_no = Label(self.root, text="Room No", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2, pady=2)
40         lbl_room_no.grid(row=1, column=0, sticky="w")

```

```

41         self.var_roomno = StringVar()
42         entry_roomno = ttk.Entry(lbl_room_no, width=17, textvariable=self.var_roomno, font=("times new roman", 19, "bold"))
43         entry_roomno.grid(row=1, column=1, sticky="w")
44
45         # ***** Room Type *****
46         lbl_room_type = Label(self.root, text="Room Type", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2, pady=2)
47         lbl_room_type.grid(row=2, column=0, sticky="w")
48
49         self.var_roomtype = StringVar()
50         entry_roomtype = ttk.Entry(lbl_room_type, width=17, textvariable=self.var_roomtype, font=("times new roman", 19, "bold"))
51         entry_roomtype.grid(row=2, column=1, sticky="w")
52
53         # ***** Room Price *****
54         lbl_room_price = Label(self.root, text="Room Price", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2, pady=2)
55         lbl_room_price.grid(row=3, column=0, sticky="w")
56
57         self.var_roomprice = StringVar()
58         entry_roomprice = ttk.Entry(lbl_room_price, width=17, textvariable=self.var_roomprice, font=("times new roman", 19, "bold"))
59         entry_roomprice.grid(row=3, column=1, sticky="w")
60
61         # ***** Room Status *****
62         lbl_room_status = Label(self.root, text="Room Status", font=("times new roman", 19, "bold"), bd=1, relief="solid", padx=2, pady=2)
63         lbl_room_status.grid(row=4, column=0, sticky="w")
64
65         self.var_roomstatus = StringVar()
66         entry_roomstatus = ttk.Entry(lbl_room_status, width=17, textvariable=self.var_roomstatus, font=("times new roman", 19, "bold"))
67         entry_roomstatus.grid(row=4, column=1, sticky="w")
68
69         # ***** Room Details *****
70         lbl_room_details = Label(self.root, text="Room Details", font=("times new roman", 22, "bold"), bd=1, relief="solid", padx=2)
71         lbl_room_details.place(x=0, y=150, width=1000, height=400)
72
73         # ***** Room Details *****
74         btn_add = Button(lbl_room_details, text="ADD", command=self.add_data, font=("times new roman", 17, "bold"), bg="black", fg="gold", width=10)
75         btn_add.grid(row=0, column=0, padx=1)
76
77         btn_update = Button(lbl_room_details, text="UPDATE", command=self.update, font=("times new roman", 17, "bold"), bg="black", fg="gold", width=10)
78         btn_update.grid(row=0, column=1, padx=1)
79
80         btn_delete = Button(lbl_room_details, text="DELETE", command=self.delete, font=("times new roman", 17, "bold"), bg="black", fg="gold", width=10)
81         btn_delete.grid(row=0, column=2, padx=1)
82
83         btn_reset = Button(lbl_room_details, text="RESET", command=self.reset_data, font=("times new roman", 17, "bold"), bg="black", fg="gold", width=10)
84         btn_reset.grid(row=0, column=3, padx=1)
85
86         # ***** Room Search *****
87         table_frame = Frame(self.root, bd=1, relief="solid", text="Show Room Details", font=("times new roman", 22, "bold"), padx=2)
88         table_frame.place(x=0, y=150, width=1000, height=400)
89
90         scroll_x = Scrollbar(table_frame, orient="horizontal")
91         scroll_y = Scrollbar(table_frame, orient="vertical")
92
93         self.room_table = ttk.Treeview(table_frame, columns=("Floor", "Roomno", "Roomtype"), xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
94
95         scroll_x.pack(side="bottom", fill="x")
96         scroll_y.pack(side="right", fill="y")

```

```

72         scroll_y.pack(side="right", fill="y")
73         scroll_x.config(command=self.room_table.xview)
74         scroll_y.config(command=self.room_table.yview)
75
76         self.room_table.heading("#Floor", text="Floor")
77         self.room_table.heading("#Roomno", text="Roomno")
78         self.room_table.heading("#Roomtype", text="Roomtype")
79
80         self.room_table["show"] = "headings"
81
82         self.room_table.column("#Floor", width=200)
83         self.room_table.column("#Roomno", width=200)
84         self.room_table.column("#Roomtype", width=200)
85         self.room_table.bind("<Button-1>", self.get_cursor)
86         self.room_table.pack(fill="both", expand=1)
87         self.fetch_data()
88
89         # ***** Add Function *****
90         def add_data(self):
91             if self.var_floor.get() == "" or self.var_roomno.get() == "":
92                 messagebox.showerror("Error", "All fields are required", parent=self.root)
93             else:
94                 try:
95                     conn = mysql.connector.connect(host="localhost", username="root", password="root", database="student1")
96                     my_cursor = conn.cursor()
97                     my_cursor.execute("insert into details values(%s,%s,%s)", (
98                         self.var_floor.get(),
99                         self.var_roomno.get(),
100                         self.var_roomtype.get()
101                     ))
102                     conn.commit()
103                     self.fetch_data()
104                     conn.close()
105                     messagebox.showinfo("Success", "New Room Added Successfully", parent=self.root)
106                 except Exception as e:
107                     messagebox.showwarning("Warning", f"Something went wrong: {str(e)}", parent=self.root)

```

```

107         # ***** Get Function *****
108         def get_data(self):
109             conn = mysql.connector.connect(host="localhost", username="root", password="root", database="student1")
110             my_cursor = conn.cursor()
111             my_cursor.execute("select * from details")
112             rows = my_cursor.fetchall()
113             if len(rows) > 0:
114                 self.room_table.delete(*self.room_table.get_children())
115                 for i in rows:
116                     self.room_table.insert("", END, values=i)
117             conn.commit()
118             conn.close()
119
120         # ***** Get Cursor *****
121         def get_cursor(self, event):
122             cursor = self.room_table.focus()
123             content = self.room_table.item(cursor)
124             row = content["values"]
125
126             self.var_floor.set(row[0])
127             self.var_roomno.set(row[1])
128             self.var_roomtype.set(row[2])
129
130         # ***** Update Function *****
131         def update(self):
132             if self.var_floor.get() == "":
133                 messagebox.showerror("Error", "Please enter floor number", parent=self.root)
134             else:
135                 conn = mysql.connector.connect(host="localhost", username="root", password="root", database="student1")
136                 my_cursor = conn.cursor()
137                 my_cursor.execute("update details set Floor=%s, Roomtype=%s where Roomno=%s", (
138                     self.var_floor.get(),
139                     self.var_roomtype.get(),
140                     self.var_roomno.get()
141                 ))
142                 conn.commit()
143                 self.fetch_data()

```

```

144 *****delete function*****
145 def delete(self):
146     delete_messagebox.askyesno("HOTEL MANAGEMENT SYSTEM","Do you want delete this Room Details",parent=self.root)
147     if delete_messagebox.askyesno():
148         conn=mysql.connector.connect(host="localhost",username="root",password="root",database="student1")
149         my_cursor=conn.cursor()
150         query="delete from details where roomno=%s"
151         value=(self.var_roomno.get(),)
152         my_cursor.execute(query,value)
153     else:
154         if not delete_messagebox.askyesno():
155             return
156         conn.commit()
157         self.fetch_data()
158         conn.close()
159 *****reset function*****
160 def reset_data(self):
161     self.var_floor.set("")
162     self.var_roomno.set("")
163     self.var_roomtype.set("")
164
165 if __name__ == "__main__":
166     root=Tk()
167     obj=detailsroom(root)
168     root.mainloop()

```

## System Architecture:

1. Presentation Layer (User Interface) built using Tkinter (Python's standard GUI library)
2. Functional Module: Customer, Room- Booking, Room- Detail
3. Databases: MySQL





