**PROJECT ON**

**PG MANAGEMENT**



**“FEEL AT HOME”**



**Submitted by** : Rakshit Agrawal

**Class** : XII B

**Roll no. :**

CERTIFICATE

CLASS: XII-B YEAR: 2020-2021

This is to certify that Investigatory Project is successfully completed by RAKSHIT AGRAWAL of Class: XII, Division: B, Roll no.: for the academic year 2020-2021 in the School Computer lab.

Head Teacher External Internal Examiner Signature: Examiner (Subject Teacher)

Date: / / 20 Department of: COMPUTER SCI.

Principal

**Acknowledgement:**

I, **RAKSHIT AGRAWAL** of class XII-B

would like to express my sincere gratitude to our computer science teacher Mr. Ashok Kr. Goel, PGT COMPUTER SCIENCE, for his vital support, guidance and encouragement – without which this project would not have come forth.

I would also like to express my gratitude to our school DAV PUBLIC SCHOOL SRESTHA VIHAR for letting us use the school laboratory.

**INDEX**

1. Brief Overview of Project
2. Need of Computerization
3. Software and Hardware requirements
4. Advantages of Project
5. Limitations of Project
6. Source Code of Project
7. Output Screens
8. Future Enhancement of Project
9. Bibliography

**PG MANAGEMENT**

BRIEF OVERVIEW OF PROJECT

The main objective of the python project on PG Management is to manage the details of rooms, residents and their payments & requirements.

The project is built at administrative as well as non-administrative end and only administrator is guaranteed the access related to residents’ data whereas non-administrators can only retrieve data to view details of vacant rooms.

Project is created with the aim to make the operations, carried out by the caretaker, easier and systematic.

The purpose of the project is to build an application program to reduce the manual work for managing the rooms, residents and payment details.

It tracks all the details about rooms, residents and payments. It also prints various reports as per input given by the user.

**INPUT DATA AND VALIDATION OF PROJECT**

1. All the fields such as are validated and do not take invalid values.
2. Many fields such as contact no., date of birth, room no. cannot accept the blank values.
3. Avoiding errors in data.
4. Controlling amount of input.

NEED OF COMPUTERISATION

Over the decades computers and PGs have developed gradually, changed with time. But nobody knew that a time will come when both these fields will complement each other so well. As a result of which, computer industry has got its new customer. Computer technology is making waves in the PG Management zone. Computers are a vital component for room bookings. New software programs continue to replace old manual skills. Those who lag in math can now breathe a little easier because manually figuring of room rents insists that knowledge. Software programs constantly evolve. A program used today may be obsolete within several years. Being trained on today's software does not guarantee it will be used when you are ready to go out into the field. Understanding calculations is timeless, as is computer competency. Software, however, shifts rapidly.

SOFTWARE AND HARDWARE REQUIREMENTS

* **SOFTWARE:**

1. Operating system (Windows 7 used): The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible for the user to use any computer or mobile device without having an operating system.
2. IDLE python (Python 3.6 32-bit used): The python software has been the backbone of our project. Without the availability of this software the project would not have started. It is an interpreted, high level and a general purpose programming language. Python has supported various modules and functions in our program which eased the making of this project.
3. MySQL: MySQL is an open source relational database management system. This application is used for various services such as data warehousing, commerce and logging applications. The software has been a major part of this project for the purpose of storing data in the form of organized records and then retrieving those records in our programming section.
4. Google chrome: The internet has been helpful in completing our project. Various details about PGs were searched and retrieved after looking for them on google.

* **HARDWARE:**

1. Processor (Intel Core i3-2328M used)
2. Ram (4.00 GB used)
3. Hard disk
4. PC/Laptop (VOSTRO 2420 used): Of course the availability of a computer or a laptop was the major requirement of our project. The work on project through various softwares stated above was carried out on a desktop.

ADVANTAGES

1. It can generate reports on room bookings, rent defaulters, residents’ requirements and many more.
2. It increases the efficiency of room bookings and makes the PG management systematic.
3. It has higher efficiency in adding, deleting and updating records.
4. It provides searching facilities on various factors and hence produces results instantly.
5. It allows the user to login as ‘Guest’ and conveniently check the details related to the available rooms keeping the residents’ information secure.
6. All the personal details of residents can be viewed and modified only by the administrator.
7. Reduces the manual work for managing the rooms, residents and payment details & reduces the time to retrieve reports related to the residents' data.

Limitations

1. All the transactions are executed in offline mode only.
2. Online transactions for bookings or other data modifications are not possible.
3. Offline reports on bookings and various other factors cannot be generated.

Source code

**DBMS: MySQL**

**Host: localhost**

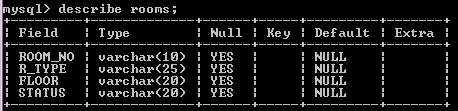
**User: root**

**Passwd: rakshit**

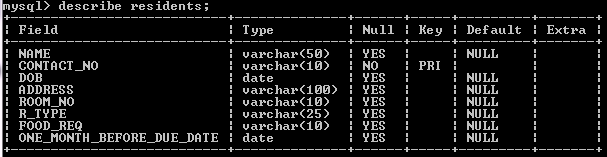
**Database: pg**

**Table Structure: (Images Below)**

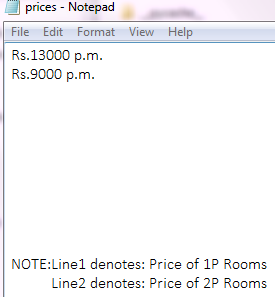
**TABLE-“ROOMS”**

****

**TABLE-“RESIDENTS”**

****

**Text File used: “prices.txt” (Image Below)**



PYTHON CODE

This python code comprises of following 5 modules and 1 text file:

* **caretaker.py:** All the functions pertaining to the operations carried out by the administrator on ‘pg’ database are defined in this module like adding records, deleting records etc.
* **guests.py:** All the functions pertaining to the operations carried out by the guest on ‘pg’ database are defined in this module such as details of unoccupied rooms: floor wise, type wise etc.
* **layout.py:** It contains the function describing the layout of the PG.
* **pg\_prog.py:** It is the main program and all the other modules are imported in this program to use their functions and combine them to form PG management system program.
* **price.py:** It contains the functions operating on the rent of the PG such as displaying & updating room rents.
* **prices.txt:** This text file stores the rent of the rooms of the PG.

CARETAKER.PY

from datetime import date

def new\_res(mycursor, mycon): #add resident

ans="yes"

while ans=="yes":

mycursor.execute("Select\* from ROOMS where STATUS<>'OCCUPIED'") #details of all unoccupied rooms

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("All rooms are occupied")

else:

print("Total no. of unoccupied rooms are=", count)

print("Details of unoccupied rooms are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

print("\n") #details of the new resident

print("Above information might be helpful in proceeding...")

print("Please enter the following details of the new resident to proceed:")

name=input("Name:")

name=name.upper()

contactno=input("Contact No.(Contact no. of 2 residents mustn't be same):")

contactno=contactno.strip()

if len(contactno)==10 and contactno.isdigit(): #contact no. is a 10 digit number

dob=input("DOB(YYYY-MM-DD):")

if dob<=str(date.today()): #dob is not in future

add=input("Address:")

roomno=input("Room no. alloted:")

roomno=roomno.upper()

rtype=input("Room type(1P/2P):")

rtype=rtype.upper()

food=input("Food Requirement(yes/no):")

food=food.upper()

dues=input("Date upto which payment has been done(YYYY-MM-DD):")

if int(dues[5:7])==1: #if block to subtract 1 month from due date

dues=str(int(dues[0:4])-1)+dues[4]+str(12)+dues[7:]

elif int(dues[5:7]) in (5, 7, 10, 12) and int(dues[8:])==31:

dues=dues[0:5]+str(int(dues[5:7])-1)+dues[7]+str(30)

elif int(dues[5:7])==3 and int(dues[8:]) in (29, 30, 31):

if int(dues[0:4])%4==0 and int(dues[8:])==29:

dues=dues[0:5]+str(int(dues[5:7])-1)+dues[7:]

elif int(dues[0:4])%4==0 and int(dues[8:]) in (30, 31):

dues=dues[0:5]+str(int(dues[5:7])-1)+dues[7]+str(29)

else:

dues=dues[0:5]+str(int(dues[5:7])-1)+dues[7]+str(28)

else:

dues=dues[0:5]+str(int(dues[5:7])-1)+dues[7:]

stat="Insert into RESIDENTS \

values('{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}')".format\

(name, contactno, dob, add, roomno, rtype, food, dues)

sq="Select\* from RESIDENTS where CONTACT\_NO='{}'".format(contactno,)

mycursor.execute(sq)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0: #entered contact no. is unique among residents

quer="Select\* from ROOMS where ROOM\_NO='{}' and R\_TYPE='{}'".format(roomno, rtype)

mycursor.execute(quer)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("INVALID INPUT")

print("The Room no. and the Room type don't match.")

else: #room exists for given room no. and room type

if data[0][3]=="OCCUPIED":

print("INVALID INPUT")

print("The Room ", roomno, " is already occupied.")

else: #room available

if rtype=="1P":

query="Update ROOMS set STATUS='OCCUPIED' where ROOM\_NO='{}' and R\_TYPE='{}'".format(roomno, rtype)

mycursor.execute(query)

mycon.commit()

print("#ROOM STATUS UPDATED#")

else: #rtype=="2P"

if data[0][3]=="VACANT":

query="Update ROOMS set STATUS='HALF OCCUPIED' where ROOM\_NO='{}' and R\_TYPE='{}'".format\

(roomno, rtype)

mycursor.execute(query)

mycon.commit()

else:

query="Update ROOMS set STATUS='OCCUPIED' where ROOM\_NO='{}' and R\_TYPE='{}'".format(roomno, rtype)

mycursor.execute(query)

mycon.commit()

print("#ROOM STATUS UPDATED#")

mycursor.execute(stat)

mycon.commit()

print("#DATA OF NEW RESIDENT ADDED#")

else:

print("INVALID INPUT")

print("A resident corresponding to the given contact no. already exists.")

else:

print("INVALID INPUT")

print("INVALID DOB")

else:

print("INVALID INPUT")

print("Contact no. must be a 10 digit number.")

ans=input("Want to add data of more residents?(yes/no):")

def t\_residents(mycursor): #all residents' details

mycursor.execute("Select\* from RESIDENTS")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("Currently no residents")

else:

print("Total no. of residents are=", count)

print("Details of all the residents are as follows:")

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

def remove\_res(mycursor, mycon): #remove resident

ans="yes"

mycursor.execute("Select\* from RESIDENTS")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("Currently no residents")

else:

while ans=="yes":

print("Please enter the following details of the resident to delete data:")

rname=input("Name:")

rname=rname.upper()

rcontactno=input("Contact No.:")

rcontactno=rcontactno.strip()

if len(rcontactno)==10 and rcontactno.isdigit(): #contact no. is a 10 digit number

rroomno=input("Room no.:")

rroomno=rroomno.upper()

querys="Select\* from RESIDENTS where NAME='{}' and CONTACT\_NO='{}' and ROOM\_NO='{}'".format\

(rname, rcontactno, rroomno)

mycursor.execute(querys)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("INVALID INPUT")

print("The Resident doesn't exist.")

else: #if resident exists

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

print("\nAbove information might be helpful in proceeding...\nPLEASE MAKE SURE THAT ALL THE DUES ARE \

CLEARED BEFORE DELETING THE DATA...")

delete=input("Are you sure that you want to delete above resident's data(yes/no):")

if delete=="yes":

stat="Delete from RESIDENTS where NAME='{}' and CONTACT\_NO='{}' and ROOM\_NO='{}'".format\

(rname, rcontactno, rroomno)

quer="Select\* from ROOMS where ROOM\_NO='{}'".format(rroomno,)

mycursor.execute(quer)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("INVALID INPUT")

print("The Room corresponding to the given room no. doesn't exist.")

else: #room exists for given room no.

if data[0][3]=="VACANT":

print("INVALID INPUT")

print("The Room ", rroomno, " is already vacant.")

else:

if data[0][1]=="1P":

query="Update ROOMS set STATUS='VACANT' where ROOM\_NO='{}'".format(rroomno,)

mycursor.execute(query)

mycon.commit()

print("#ROOM STATUS UPDATED#")

else: #rtype=="2P"

if data[0][3]=="OCCUPIED":

query="Update ROOMS set STATUS='HALF OCCUPIED' where ROOM\_NO='{}'".format(rroomno,)

mycursor.execute(query)

mycon.commit()

else:

query="Update ROOMS set STATUS='VACANT' where ROOM\_NO='{}'".format(rroomno,)

mycursor.execute(query)

mycon.commit()

print("#ROOM STATUS UPDATED#")

mycursor.execute(stat)

mycon.commit()

print("#DATA OF RESIDENT DELETED#")

elif delete=="no":

print("OPERATION CANCELLED")

else:

print("INVALID INPUT")

else:

print("INVALID INPUT")

print("Contact no. must be a 10 digit number.")

ans=input("Want to delete data of more residents?(yes/no):")

def dues(mycursor, mycon): #rent defaulters

pdate=date.today()

pdate=str(pdate)

if int(pdate[5:7])==1: #if block to subtract 1 month from due date

prdate=str(int(pdate[0:4])-1)+pdate[4]+str(12)+pdate[7:]

elif int(pdate[5:7]) in (5, 7, 10, 12) and int(pdate[8:])==31:

prdate=pdate[0:5]+str(int(pdate[5:7])-1)+pdate[7]+str(30)

elif int(pdate[5:7])==3 and int(pdate[8:]) in (29, 30, 31):

if int(pdate[0:4])%4==0 and int(pdate[8:])==29:

prdate=pdate[0:5]+str(int(pdate[5:7])-1)+pdate[7:]

elif int(pdate[0:4])%4==0 and int(pdate[8:]) in (30, 31):

prdate=pdate[0:5]+str(int(pdate[5:7])-1)+pdate[7]+str(29)

else:

prdate=pdate[0:5]+str(int(pdate[5:7])-1)+pdate[7]+str(28)

else:

prdate=pdate[0:5]+str(int(pdate[5:7])-1)+pdate[7:]

statement="Select\* from RESIDENTS where ONE\_MONTH\_BEFORE\_DUE\_DATE<='{}' order by \

ONE\_MONTH\_BEFORE\_DUE\_DATE asc".format(prdate,)

mycursor.execute(statement)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("All dues are clear")

else:

print("RENT PAYMENT POLICY: 1.Security deposit is 1 month rent \n 2.Rent must be paid monthly\n")

print("Total no. of residents with pending dues are=", count)

print("Details of all residents with pending dues are as follows:")

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

if int(row[7][5:7])==12: #if block to find due date

dd=str(int(row[7][0:4])+1)+row[7][4]+"01"+row[7][7:]

elif int(row[7][5:7]) in (3, 5, 8, 10) and int(row[7][8:])==31:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(30)

elif int(row[7][5:7])==1 and int(row[7][8:]) in (29, 30, 31):

if int(row[7][0:4])%4==0 and int(pdate[8:])==29:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7:]

elif int(pdate[0:4])%4==0 and int(pdate[8:]) in (30, 31):

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(29)

else:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(28)

else:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7:]

print("=>", row[0], " cleared all the dues only till ", dd, ".", end="")

print(" Therefore, according to the RENT PAYMENT POLICY, rent that is due, is from ", dd, ".")

answ=input("Want to clear any dues?(yes/no):")

if answ=="yes":

clear\_dues(mycursor, mycon)

elif answ=="no":

nothing="nothing"

else:

print("INVALID INPUT")

def clear\_dues(mycursor, mycon): #clear dues

mycursor.execute("Select\* from RESIDENTS")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("Currently no residents")

else:

print("Please enter the following details of the resident to clear his dues:")

rname=input("Name:")

rname=rname.upper()

rcontactno=input("Contact No.:")

rcontactno=rcontactno.strip()

if len(rcontactno)==10 and rcontactno.isdigit(): #contact no. is a 10 digit number

rroomno=input("Room no.: ")

rroomno=rroomno.upper()

querys="Select\* from RESIDENTS where NAME='{}' and CONTACT\_NO='{}' and ROOM\_NO='{}'".format(rname, rcontactno, rroomno)

mycursor.execute(querys)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("INVALID INPUT")

print("The Resident doesn't exist.")

else: #if resident exists

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

if int(row[7][5:7])==12: #if block to find due date

dd=str(int(row[7][0:4])+1)+row[7][4]+"01"+row[7][7:]

elif int(row[7][5:7]) in (3, 5, 8, 10) and int(row[7][8:])==31:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(30)

elif int(row[7][5:7])==1 and int(row[7][8:]) in (29, 30, 31):

if int(row[7][0:4])%4==0 and int(pdate[8:])==29:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7:]

elif int(pdate[0:4])%4==0 and int(pdate[8:]) in (30, 31):

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(29)

else:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7]+str(28)

else:

dd=row[7][0:5]+str(int(row[7][5:7])+1)+row[7][7:]

print("=>", row[0], " cleared all the dues till ", dd, ".")

print("Above information might be helpful in proceeding...")

dates=input("Please enter a new date upto which all the dues are cleared(YYYY-MM-DD):")

if int(dates[5:7])==1: #if block to subtract 1 month from new due date

dates=str(int(dates[0:4])-1)+dates[4]+str(12)+dates[7:]

elif int(dates[5:7]) in (5, 7, 10, 12) and int(dates[8:])==31:

dates=dates[0:5]+str(int(dates[5:7])-1)+dates[7]+str(30)

elif int(dates[5:7])==3 and int(dates[8:]) in (29, 30, 31):

if int(dates[0:4])%4==0 and int(dates[8:])==29:

dates=dates[0:5]+str(int(dates[5:7])-1)+dates[7:]

elif int(pdate[0:4])%4==0 and int(pdate[8:]) in (30, 31):

dates=dates[0:5]+str(int(dates[5:7])-1)+dates[7]+str(29)

else:

dates=dates[0:5]+str(int(dates[5:7])-1)+dates[7]+str(28)

else:

dates=dates[0:5]+str(int(dates[5:7])-1)+dates[7:]

queryss="Update RESIDENTS set ONE\_MONTH\_BEFORE\_DUE\_DATE='{}' where NAME='{}' and CONTACT\_NO='{}' \

and ROOM\_NO='{}'".format(dates, rname, rcontactno, rroomno)

mycursor.execute(queryss)

mycon.commit()

print("#DUES CLEARED#")

else:

print("INVALID INPUT")

print("Contact no. must be a 10 digit number.")

def food\_residents(mycursor): #residents availing food facility

mycursor.execute("Select\* from RESIDENTS where FOOD\_REQ='YES'")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Food facility is not being availed by anyone")

else:

print("Total no. of residents availing food facility are=", count)

print("Details of residents availing food facility are as follows:")

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

def ch\_food(mycursor, mycon): #change food req.

mycursor.execute("Select\* from RESIDENTS")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("Currently no residents")

else:

print("Please enter the following details of the resident to change food requirement:")

rname=input("Name:")

rname=rname.upper()

rcontactno=input("Contact No.:")

rcontactno=rcontactno.strip()

if len(rcontactno)==10 and rcontactno.isdigit(): #contact no. is a 10 digit number

rroomno=input("Room no.: ")

rroomno=rroomno.upper()

querys="Select\* from RESIDENTS where NAME='{}' and CONTACT\_NO='{}' and ROOM\_NO='{}'".format(rname, rcontactno, rroomno)

mycursor.execute(querys)

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("INVALID INPUT")

print("The Resident doesn't exist.")

else: #if resident exists

print("\n\

('NAME', 'CONTACT NO.', 'DOB', 'ADDRESS', 'ROOM NO.', 'ROOM TYPE', 'FOOD REQ.', 'ONE\_MONTH\_BEFORE\_DUE\_DATE')\n")

for row in data:

row=list(row)

row[7]=str(row[7])

row[2]=str(row[2])

row=tuple(row)

print(row)

if row[6]=="YES":

print("=>", row[0], " is availing food facility.")

else:

print(row[0], " is not availing food facility.")

print("Above information might be helpful in proceeding...")

foodreq=input("Please enter your choice, Food requirement(yes/no):")

foodreq=foodreq.upper()

queryss="Update RESIDENTS set FOOD\_REQ='{}' where NAME='{}' and CONTACT\_NO='{}' and ROOM\_NO='{}'".format\

(foodreq, rname, rcontactno, rroomno)

mycursor.execute(queryss)

mycon.commit()

print("#FOOD REQUIREMENT UPDATED#")

else:

print("INVALID INPUT")

print("Contact no. must be a 10 digit number.")

GUESTS.PY

def t\_rooms(mycursor): #all rooms' details

mycursor.execute("Select\* from ROOMS")

data=mycursor.fetchall()

count=mycursor.rowcount

print("Total no. of rooms are=", count)

print("Details of rooms are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

def unocc\_rooms(mycursor): #all unocc. rooms

mycursor.execute("Select\* from ROOMS where STATUS<>'OCCUPIED'")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("All rooms are occupied")

else:

print("Total no. of unoccupied rooms are=", count)

print("Details of unoccupied rooms are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

def unocc\_1prooms(mycursor): #all unocc. 1P rooms

mycursor.execute("Select\* from ROOMS where STATUS<>'OCCUPIED' and R\_TYPE='1P'")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("All 1P rooms are occupied")

else:

print("Total no. of 1P unoccupied rooms are=", count)

print("Details of 1P unoccupied rooms are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

def unocc\_2prooms(mycursor): #all unocc. 2P rooms

mycursor.execute("Select\* from ROOMS where STATUS<>'OCCUPIED' and R\_TYPE='2P'")

data=mycursor.fetchall()

count=mycursor.rowcount

if count==0:

print("Sorry!!")

print("All 2P rooms are occupied")

else:

print("Total no. of 2P unoccupied rooms are=", count)

print("Details of 2P unoccupied rooms are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

def F0\_rooms(mycursor, y): #unocc. rooms(floorwise)

st="Select\* from ROOMS where STATUS<>'OCCUPIED' and FLOOR='{}'".format(y,)

mycursor.execute(st)

data=mycursor.fetchall()

count=mycursor.rowcount

z=y.lower()

if count==0:

print("Sorry!!")

print("All rooms on ", z, " floor are occupied")

else:

print("Total no. of unoccupied rooms on ", z, " floor are=", count)

print("Details of unoccupied rooms on ", z, " floor are as follows:")

print("\n\

('ROOM NO.', 'ROOM TYPE', 'FLOOR', 'STATUS')\n")

for row in data:

print(row)

LAYOUT.PY

def pgmap(): #Displays floor map

print("Ground Floor=>") #ground floor map

print("\

\t\t\t+--------+------------+--------+\n\

\t\t\t| | | |\n\

\t\t\t| | STAIRCASE | |\n\

\t\t\t| G02 +------------+ G01 |\n\

\t\t\t| | | |\n\

\t\t\t+--------+ +--------+\n\

\t\t\t| |\n\

\t\t\t E +------+ |\n\

\t\t\t| | R | |\n\

\t\t\t+-----------+------+-----------+")

#

#

print("Other Floors=>") #other floors' map

print("\

\t\t\t+--------+-----------+--------+\n\

\t\t\t| | | |\n\

\t\t\t| | STAIRCASE | |\n\

\t\t\t| F02 +-----------+ F01 |\n\

\t\t\t| | | |\n\

\t\t\t+--------+ +--------+\n\

\t\t\t| | | |\n\

\t\t\t| +-----+-----+ |\n\

\t\t\t| F04(2P) | F03(2P) |\n\

\t\t\t| | |\n\

\t\t\t+--------------+--------------+\n")

print("where,R denotes RECEPTION\n\

E denotes ENTRY\n\

and F denotes Floor number(where, F=1/2/3)")

Pg**\_**PROG.PY

#PROJECT DETAILS

#COMPUTER CLASS XII PROJECT

#TOPIC: PG MANAGEMENT

#BY: RAKSHIT AGRAWAL(No Project Partner)

# XII B

import price

import layout

import caretaker

import guests

import mysql.connector as ms

from getpass import getpass

mycon=ms.connect(host='localhost', user='root', passwd='rakshit',charset='utf8', database='PG')

mycursor=mycon.cursor()

#Details of ROOMS already in SQL Database

#PG has Ground and 3 extra floors

#ONE\_MONTH\_BEFORE\_DUE\_DATE column denotes the date that is one month prior to due date.

#Residents must pay the rent of the month before staying for that month.

print("\t\t\t\t", 27\*"\*\*")

print("\t\t\t\t\t WELCOME TO COSMIC BOYS PG")

print("\t\t\t\t", 27\*"\*\*")

print("\n#FOOD & AMENITIES:\n We offer you a clean, hygienic, spacious and secure accommodation along with the \

following amenities:\n\n")

print(" Air conditioner Internet Kitchen 2-Wheeler Parking\n\n\

24hrs Hot Water CCTV Security Food Facility [No extra charges]")

print("\n\n#ROOMS:\n Wide variety of well ventilated and clean rooms, offering full privacy are:")

print("\n 1P Rooms: Single occupancy rooms[", end="")

print(price.r\_price1p(), end="")

print("]")

print("\n 2P Rooms: Double occupancy rooms[", end="")

print(price.r\_price2p(), end="")

print("]")

print(60\*"\_\_")

print("\n RENT PAYMENT POLICY: 1.Security deposit is 1 month rent \n 2.Rent must be paid monthly \n \

3.All payments are non-refundable")

print(60\*"\_\_")

print("\n", end="")

enter=input("Press ENTER to continue...\n")

#

#MAIN PROGRAM IS DIVIDED INTO TWO PARTS

#

def part1(): #For user as guest

print("Choose from the following options to get to know about:")

print("1.FLOORWISE LAYOUT of PG")

print("2.Details of all rooms")

print("3.Details of unoccupied rooms w.r.t. the room type")

print("4.Details of unoccupied rooms w.r.t. the floor")

cho=int(input("Enter your choice(1/2/3/4)=>"))

if cho==1: #floorwise layout

layout.pgmap()

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif cho==2: #all rooms' details

guests.t\_rooms(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif cho==3: #unocc. rooms(typewise)

print("Choose from the following options to get to know about:")

print("1.Details of all unoccupied rooms")

print("2.Details of all 1P unoccupied rooms")

print("3.Details of all 2P unoccupied rooms")

choi=int(input("Enter your choice(1/2/3)=>"))

if choi==1: #all unocc. rooms

guests.unocc\_rooms(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choi==2: #all unocc. 1P rooms

guests.unocc\_1prooms(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choi==3: #all unocc. 2P rooms

guests.unocc\_2prooms(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif cho==4: #unocc. rooms(floorwise)

x=int(input("Please specify the floor whose rooms' details you want to know(0/1/2/3):"))

if x in (0,1,2,3):

if x==0:

y="GROUND"

elif x==1:

y="FIRST"

elif x==2:

y="SECOND"

elif x==3:

y="THIRD"

guests.F0\_rooms(mycursor, y)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Administrator:")

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

elif goal==2:

part1()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

else:

print("INVALID INPUT")

else:

print("INVALID INPUT")

def part2(): #For user as administrator

print("Choose from the following options to:")

print("1.Get Details of all residents")

print("2.Add details of a new resident")

print("3.Remove details of a resident")

print("4.Get Details of all residents with pending dues(for >=1 month)")

print("5.Clear dues of a resident")

print("6.Get Details of all residents availing food facility")

print("7.Change food requirement of a resident")

print("8.Change rooms' rent")

choice=int(input("Enter your choice(1/2/3/4/5/6/7/8)=>"))

if choice==1: #residents' details

caretaker.t\_residents(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==2: #add resident

caretaker.new\_res(mycursor, mycon)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==3: #remove resident

caretaker.remove\_res(mycursor, mycon)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==4: #rent defaulters

caretaker.dues(mycursor, mycon)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==5: #clear dues

caretaker.clear\_dues(mycursor, mycon)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==6: #residents availing food facility

caretaker.food\_residents(mycursor)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==7: #change food req.

caretaker.ch\_food(mycursor, mycon)

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

elif choice==8: #change rent

price.ch\_rent()

print("\n\nPress[1 to SWITCH USER], [2 for BACK TO MENU], [3 for EXIT]") #switch user

goal=int(input("Enter your choice(1/2/3)=>"))

if goal==1:

print("\nLogging in as Guest:")

part1()

elif goal==2:

part2()

elif goal==3:

print("\t\t\t\t\t\t---THANKYOU---")

else:

print("INVALID INPUT")

else:

print("INVALID INPUT")

#

#NOTE: Guest is only given rights to access details of unoccupied rooms by security point of view

#

#ALL THE MODULES AND FUNCTIONS ARE COMBINED IN THE FOLLOWING PROGRAM

y=""

print("Login as: 1.Guest \n 2.Administrator\n")

ch=int(input("Enter your choice(1/2)=>"))

if ch==1:

part1()

elif ch==2:

user=input("USERNAME:") #USERNAME:rakshit

password=getpass("PASSWORD:") #PASSWORD:rakshit

if user=="rakshit" and password=="rakshit":

part2()

else:

print("INVALID INPUT")

print("Username and password do not match.")

else:

print("INVALID INPUT")

mycon.close()

PRICE.PY

def r\_price1p(): #Displays price of 1P type rooms

myfile=open("prices.txt", "r")

line1=myfile.readline()

return(line1.strip())

myfile.close()

def r\_price2p(): #Displays price of 2P type rooms

myfile=open("prices.txt", "r")

line1=myfile.readline()

line2=myfile.readline()

return(line2.rstrip())

myfile.close()

def ch\_rent(): #Changes rent of rooms

file=open("prices.txt", "r")

data=file.readlines()

file.close()

s1="Rs."

s2=" p.m.\nRs."

s3=" p.m.\n\n\

NOTE:Line1 denotes: Price of 1P Rooms\n \

Line2 denotes: Price of 2P Rooms"

print("Please specify the rent you want to update:")

print("Press[1 for 1P rooms], [2 for 2P rooms]")

option=int(input("Enter your choice(1/2)=>"))

if option==1:

rent=input("Please enter the new rent:")

rent=rent.strip()

if rent.isdigit():

myfile=open("prices.txt", "w")

data[0]=s1+rent+s2[0:7]

myfile.writelines(data)

myfile.close()

print("#RENT UPDATED#")

else:

print("INVALID INPUT")

print("Enter only numeric value(for rent)")

elif option==2:

rent=input("Please enter the new rent:")

rent=rent.strip()

if rent.isdigit():

myfile=open("prices.txt", "w")

data[1]=s1+rent+s3[0:7]

myfile.writelines(data)

myfile.close()

print("#RENT UPDATED#")

else:

print("INVALID INPUT")

print("Enter only numeric value(for rent)")

else:

print("INVALID INPUT")

'''

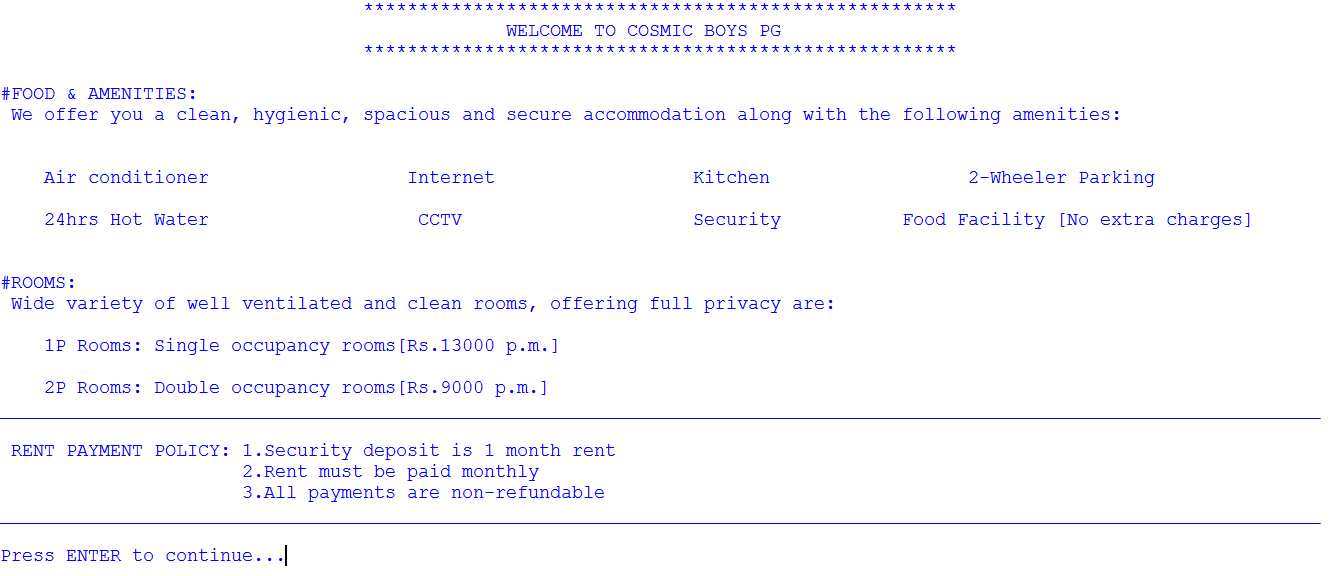
Rs.13000 p.m.

Rs.9000 p.m.

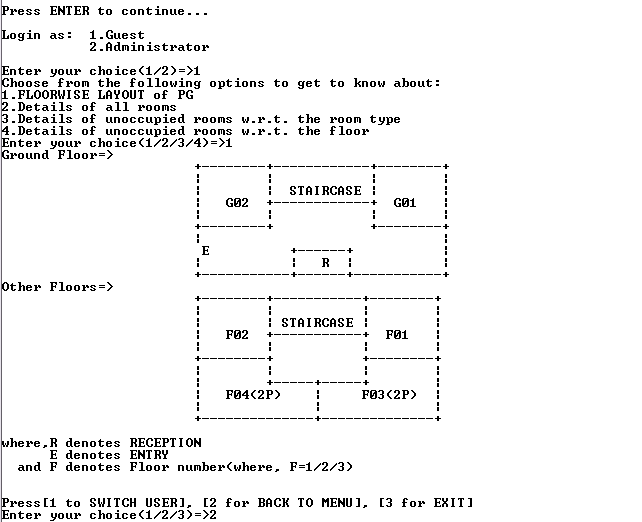
NOTE:Line1 denotes: Price of 1P Rooms

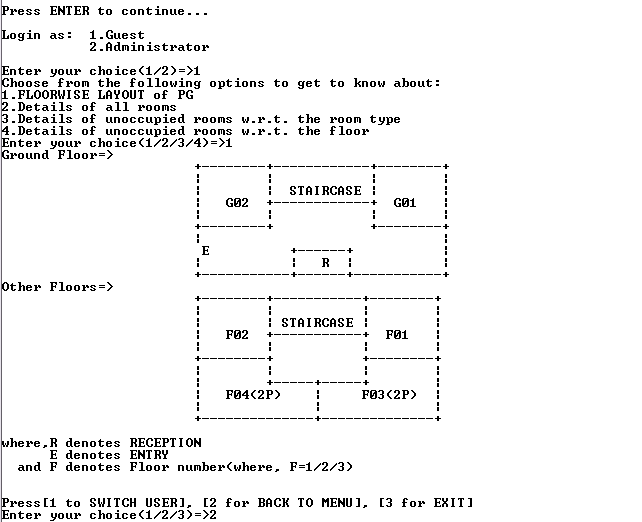
Line2 denotes: Price of 2P Rooms'''

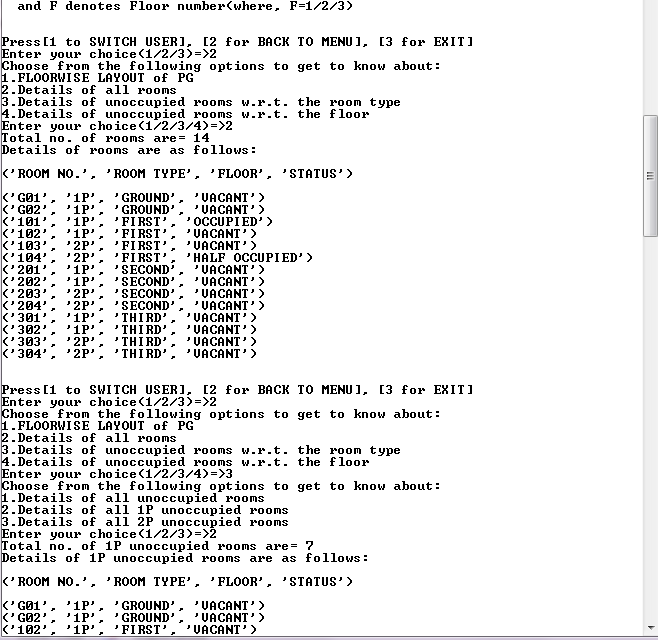
OUTPUT SCREENS



[Logging in as Guest:]



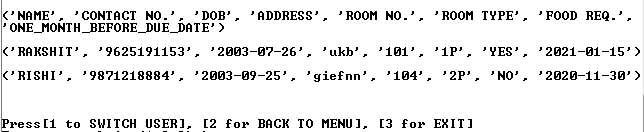


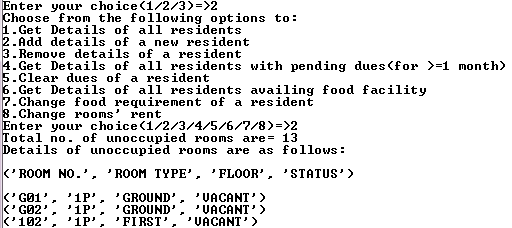


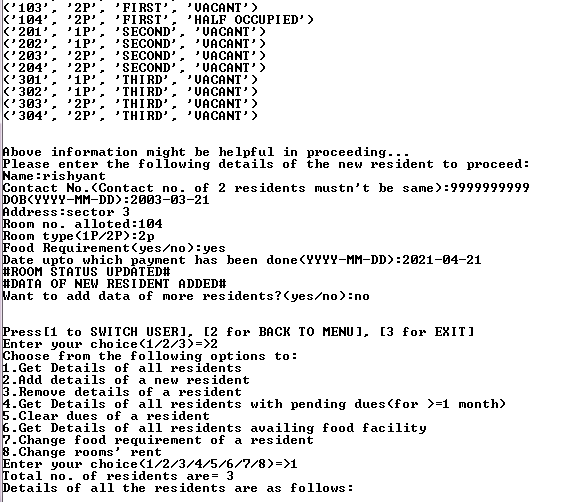


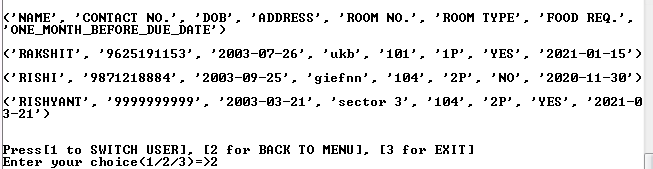
[Switching user to Administrator:]

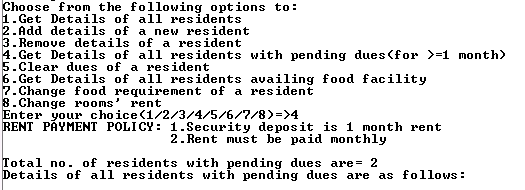


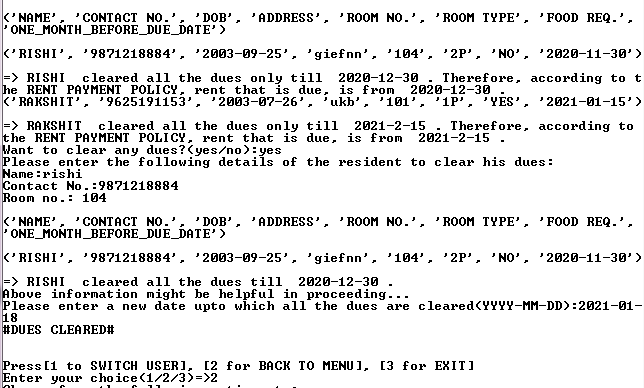


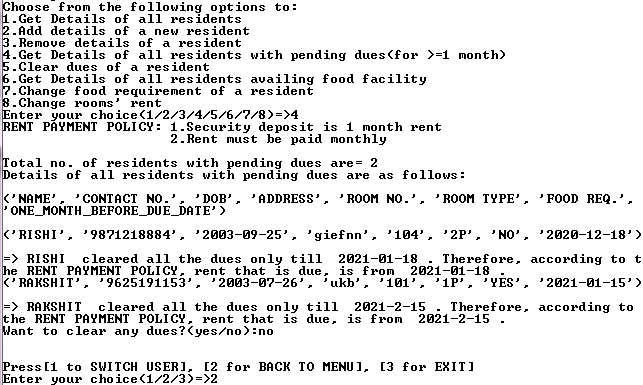


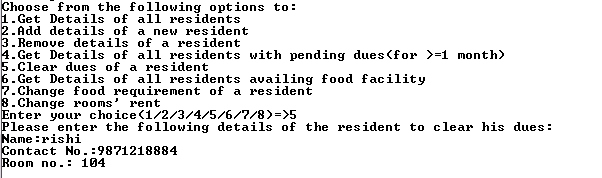


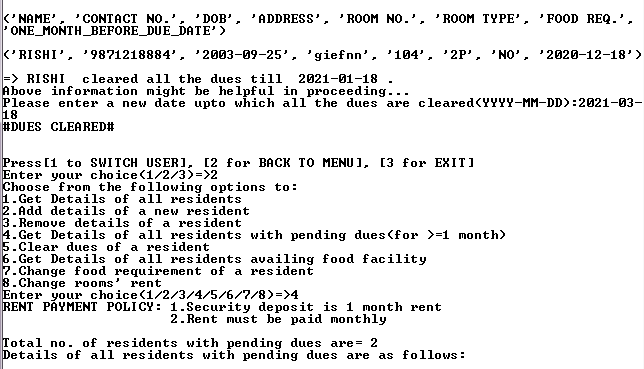


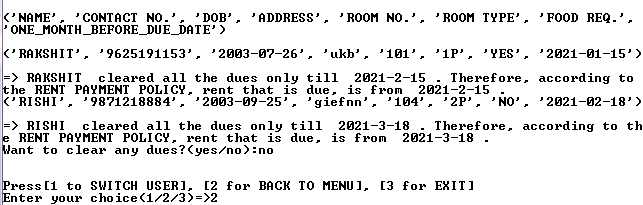


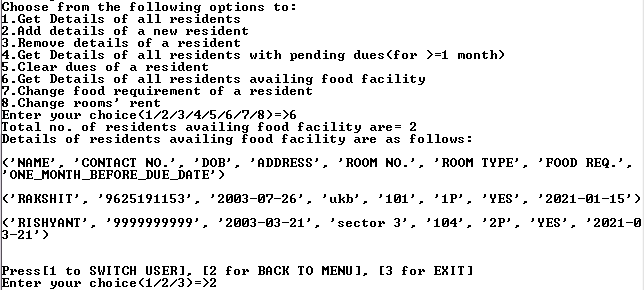


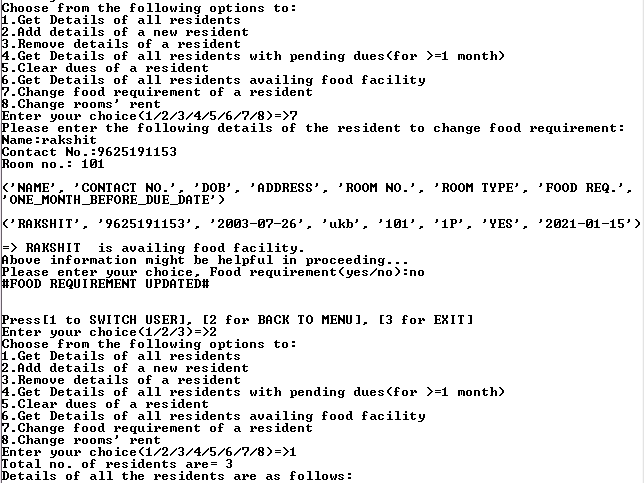


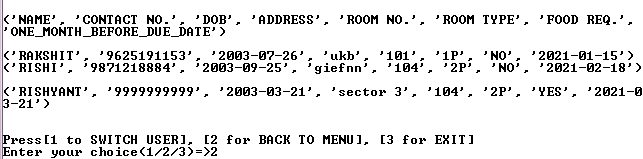


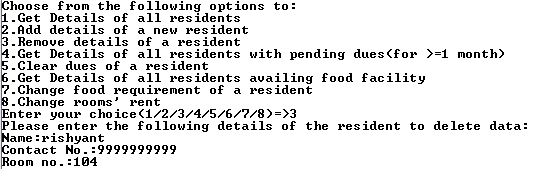


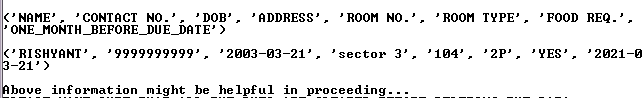


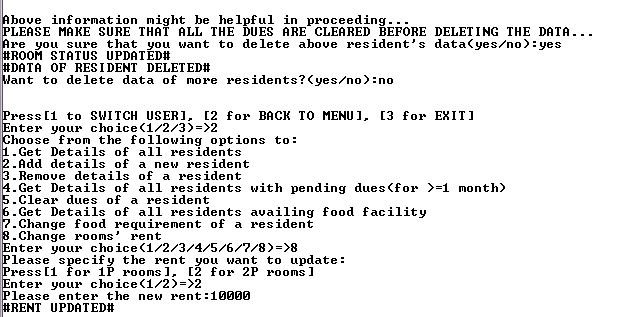


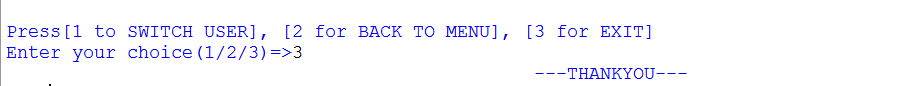




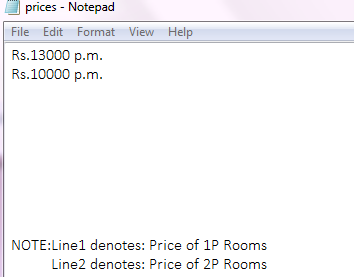








[‘prices.txt’ after Updating Rent:]



Future enhancements

1. The solutions are given as a proposal. The suggestion can be revised on user request and optimal changes can be made.
2. So on the whole, system analysis is done to improve the system performance by monitoring it and obtaining the best throughput possible from it. Therefore system analysis plays a crucial role in designing any system.
3. By some enhancements, this program can serve as an interface which can be accessed worldwide to carry out room bookings from any place.
4. The program can be further developed so that people can easily make room bookings just from a single system. On the other hand, it also removes an extra burden from the caretaker as most of the bookings are done by people using this service to book rooms at the comfort level of their home instead of making bookings from the counters.

BIBLIOGRAPHY

1. <http://www.google.com/>

2. Computer science with python

by Sumita Arora