**HOTEL MANAGEMENT SYSTEM**

**ABSTRACT**

Our project basically manages the hotel process of a MNC, providing an interface to the user to order food in a more easy way. At the front end we have used Python 3 and at the back end SQLite server. The project proceeds through a sequence of well-designed forms provided with validations to ensure consistency, reliability and most importantly correctness of information fed into the database.

**INTRODUCTION**

Welcome to newly designed application, Hotel management is a faster, cleaner and a tad more personal website, specially designed to make your booking experience better.

Admin may view, add, modify and remove the menu details. Admin may also view, add or remove bookings and shows as per needs. You need to register a new user whenever you have first visited or site then for future it will be stored in our database permanently and you can order you bookings at any time you want with this username and password. Registered Users are allowed to login and view the contents of any menu show at any time and may book any dish as needed. The program automatically books the seats for the tables and the booking is stored in the database securely. Registered Users are also allowed to Update their profile details except username and view the bookings of the user.

**SOURCE CODE :**

**# Database\_Reboot.py**

import sqlite3

import os

DRPASS = '1234'

print('-----Database Reboot------')

pas=input('Password : ')

if pas==DRPASS:

print('Rebooting Database Will Delete existing Database...')

ch=input('Do you wish to continue ? (Y) :')

if ch=='Y':

try:

print('Deleting Existing Database...')

os.remove("database.db")

print('Database Deletion Successful!')

except:

pass

print('Creating new Database...')

conn = sqlite3.connect('database.db')

print('Database Creation Successful!')

print('Creating Tables...')

conn.execute('''CREATE TABLE users

(name TEXT NOT NULL,

username TEXT PRIMARY KEY NOT NULL,

password TEXT NOT NULL,

mobilenum NUMBER NOT NULL);''')

print('Created Users Table...')

conn.execute('''CREATE TABLE movies

(movieid TEXT PRIMARY KEY NOT NULL,

moviename TEXT NOT NULL,

director TEXT NOT NULL,

duration NUMBER NOT NULL);''')

print('Created Menu Table...')

conn.execute('''CREATE TABLE screen

(screennum NUMBER PRIMARY KEY NOT NULL,

floor NUMBER NOT NULL,

capacity NUMBER NOT NULL,

CHECK (capacity>0));''')

print('Created Branch Table...')

conn.execute('''CREATE TABLE shows

(showid TEXT PRIMARY KEY NOT NULL,

movieid TEXT NOT NULL,

showtime DATETIME NOT NULL,

screenno NUMBER NOT NULL,

ava\_seats NUMBER NOT NULL,

CHECK (ava\_seats>=0));''')

print('Created Booking Table...')

conn.execute('''CREATE TABLE tickets

(ticketid NUMBER PRIMARY KEY NOT NULL,

username TEXT NOT NULL,

showid TEXT NOT NULL,

seats NUMBER NOT NULL);''')

print('Created tables Table...')

print('Table Creation Successful!')

conn.commit()

conn.close()

else:

print('Exiting!')

else:

print('Access Denied!')

os.system("pause")

**# Admin.py**

import sqlite3

from tkinter import \*

from tkinter import messagebox

def login():

username=input('Enter Admin Username : ')

password=input('Enter Password : ')

if username=='Admin' and password=='password':

print('Access Granted')

a=Admin()

a.start()

else :

print('Access Denied')

class Admin(Tk):

def start(self):

self.title("Online Hotel management")

self.adminHome()

self.mainloop()

def adminHome(self):

self.clear()

B1=Button(self,text="MENU",command= self.Movies)

B1.pack()

B2=Button(self,text="BRANCH",command=self.Screens)

B2.pack()

B3=Button(self,text="BOOKING",command=self.Shows)

B3.pack()

B4=Button(self,text='Logout',command=self.logout)

B4.pack()

def Movies(self):

self.clear()

B=Button(self,text='Back',command=self.adminHome)

B.grid()

L=Label(self,text = '--MENU--')

L.grid()

L1=Label(self,text = 'MenuID : ')

L1.grid(row=2,sticky="W")

E1=Entry(self)

E1.grid(row=2,sticky="E")

B1=Button(self,text="Remove MENU",command= lambda:self.removeMovie(E1.get()))

B1.grid()

L3=Label(self,text = "MenuID : ")

L3.grid(row=5,sticky="W")

E3=Entry(self)

E3.grid(row=5,sticky="E")

L4=Label(self,text = "MenuName : ")

L4.grid(row=6,sticky="W")

E4=Entry(self)

E4.grid(row=6,sticky="E")

L5=Label(self,text = "Cuisine : ")

L5.grid(row=7,sticky="W")

E5=Entry(self)

E5.grid(row=7,sticky="E")

L6=Label(self,text = "Prep time : ")

L6.grid(row=8,sticky="W")

vcmd = (self.register(self.onValidate),'%d','%i','%P','%s','%S','%v','%V','%W')

E6=Entry(self,validate='key',validatecommand=vcmd)

E6.grid(row=8,sticky="E")

B2=Button(self,text="Add Menu",command=lambda: self.addMovie(E3.get(),E4.get(),E5.get(),E6.get()))

B2.grid(row=9,sticky="W")

B3=Button(self,text="Edit Menu",command=lambda: self.editMovie(E3.get(),E4.get(),E5.get(),E6.get()))

B3.grid(row=9,sticky="E")

L2=Label(self,text ='--Menu List--')

L2.grid()

conn = sqlite3.connect('database.db')

tickets=conn.execute('SELECT movieid,moviename,director,duration FROM movies ORDER BY moviename')

msg='MenuID'+'\tMenuName'+'\tCuisine'+'\tPrep time'

M=Label( self, text=msg,relief = RAISED )

M.grid(sticky="W")

for row in tickets:

msg=str(row[0])+'\t'+str(row[1])+'\t\t'+str(row[2])+'\t'+str(row[3])

M=Label( self, text=msg )

M.grid(sticky="W")

conn.close()

def removeMovie(self,movieID):

conn = sqlite3.connect('database.db')

conn.execute("DELETE FROM movies WHERE movieid LIKE ?;",[movieID])

conn.execute("DELETE FROM shows WHERE movieid LIKE ?;",[movieID])

messagebox.showinfo('done',"Menu and Bookings Removed!")

conn.commit()

conn.close()

self.Movies()

def editMovie(self,movieID,movieName,director,duration):

conn = sqlite3.connect('database.db')

conn.execute("UPDATE movies SET moviename=?,director=?,duration=? WHERE movieID LIKE ?;",[movieName,director,duration,movieID])

messagebox.showinfo('done',"Menu Details Updated!")

conn.commit()

conn.close()

self.Movies()

def addMovie(self,movieID,movieName,director,duration):

conn = sqlite3.connect('database.db')

try:

conn.execute("INSERT INTO movies VALUES (?,?,?,?);",[movieID,movieName,director,duration])

except :

messagebox.showinfo('error','MenuID Already Exists!')

else:

messagebox.showinfo('done',"Menu Added!")

conn.commit()

self.Movies()

conn.close()

def Screens(self):

self.clear()

B=Button(self,text='Back',command=self.adminHome)

B.grid()

L=Label(self,text = '--Branch--')

L.grid()

vcmd = (self.register(self.onValidate),'%d','%i','%P','%s','%S','%v','%V','%W')

L1=Label(self,text = 'BranchNum : ')

L1.grid(row=2,sticky="W")

E1=Entry(self)

E1.grid(row=2,sticky="E")

B1=Button(self,text="Remove Branch",command= lambda:self.removeScreen(E1.get()))

B1.grid()

L3=Label(self,text = "BranchNum : ")

L3.grid(row=5,sticky="W")

E3=Entry(self,validate='key',validatecommand=vcmd)

E3.grid(row=5,sticky="E")

L4=Label(self,text = "Floor : ")

L4.grid(row=6,sticky="W")

E4=Entry(self,validate='key',validatecommand=vcmd)

E4.grid(row=6,sticky="E")

L5=Label(self,text = "Capacity : ")

L5.grid(row=7,sticky="W")

E5=Entry(self,validate='key',validatecommand=vcmd)

E5.grid(row=7,sticky="E")

B2=Button(self,text="Add Branch",command=lambda: self.addScreen(E3.get(),E4.get(),E5.get()))

B2.grid()

L2=Label(self,text ='--Branch List--')

L2.grid()

conn = sqlite3.connect('database.db')

tickets=conn.execute('SELECT screennum,floor,capacity FROM screen ORDER BY screennum')

msg='BranchNum'+'\tFloor'+'\tCapacity'

M=Label( self, text=msg,relief = RAISED )

M.grid(sticky="W")

for row in tickets:

msg=str(row[0])+'\t\t'+str(row[1])+'\t'+str(row[2])

M=Label( self, text=msg )

M.grid(sticky="W")

conn.close()

def removeScreen(self,screenNum):

conn = sqlite3.connect('database.db')

conn.execute("DELETE FROM screen WHERE screennum LIKE ?;",[screenNum])

messagebox.showinfo('done',"Branch Removed!")

conn.commit()

conn.close()

self.Screens()

def addScreen(self,screenNum,floor,capacity):

conn = sqlite3.connect('database.db')

try:

conn.execute("INSERT INTO screen VALUES (?,?,?);",[screenNum,floor,capacity])

except :

messagebox.showinfo('error','BranchNum Already Exists!')

else:

messagebox.showinfo('done',"Branch Added!")

conn.commit()

self.Screens()

conn.close()

def Shows(self):

self.clear()

B=Button(self,text='Back',command=self.adminHome)

B.grid()

L=Label(self,text = '--Bookings--')

L.grid()

L1=Label(self,text = 'BookingID : ')

L1.grid(row=2,sticky="W")

E1=Entry(self)

E1.grid(row=2,sticky="E")

B1=Button(self,text="Remove Booking",command= lambda:self.removeShow(E1.get()))

B1.grid()

L3=Label(self,text = "BookingID : ")

L3.grid(row=5,sticky="W")

E3=Entry(self)

E3.grid(row=5,sticky="E")

L4=Label(self,text = "DishID : ")

L4.grid(row=6,sticky="W")

E4=Entry(self)

E4.grid(row=6,sticky="E")

L5=Label(self,text = "Session : ")

L5.grid(row=7,sticky="W")

E5=Entry(self)

E5.grid(row=7,sticky="E")

L6=Label(self,text = "BranchNum : ")

L6.grid(row=8,sticky="W")

vcmd = (self.register(self.onValidate),'%d','%i','%P','%s','%S','%v','%V','%W')

E6=Entry(self,validate='key',validatecommand=vcmd)

E6.grid(row=8,sticky="E")

B2=Button(self,text="Add Booking",command=lambda: self.addShow(E3.get(),E4.get(),E5.get(),E6.get()))

B2.grid(row=9)

L2=Label(self,text ='--Bookings List--')

L2.grid()

conn = sqlite3.connect('database.db')

tickets=conn.execute('''SELECT showid,moviename,showtime,screenno,ava\_seats FROM shows

INNER JOIN movies ON movies.movieid=shows.movieid

ORDER BY moviename''')

msg='BookingID'+'\tDishName'+'\tSession'+'\tBranch'+'\tAvailableSeats'

M=Label( self, text=msg,relief = RAISED )

M.grid(sticky="W")

for row in tickets:

msg=str(row[0])+'\t\t'+str(row[1])+'\t\t'+str(row[2])+'\t'+str(row[3])+'\t'+str(row[4])

M=Label( self, text=msg )

M.grid(sticky="W")

conn.close()

def removeShow(self,showID):

conn = sqlite3.connect('database.db')

conn.execute("DELETE FROM shows WHERE showid LIKE ?;",[showID])

messagebox.showinfo('done',"Booking Removed!")

conn.commit()

conn.close()

self.Shows()

def addShow(self,showID,movieID,showTime,screenNum):

conn = sqlite3.connect('database.db')

obj=conn.execute("SELECT capacity FROM screen WHERE screennum=?",[screenNum])

capacity=obj.fetchone()

if capacity is None:

messagebox.showinfo('error','Invalid Branch!')

else:

capacity=capacity[0]

try:

conn.execute("INSERT INTO shows VALUES (?,?,?,?,?);",[showID,movieID,showTime,screenNum,capacity])

except:

messagebox.showinfo('error','BookingID Already Exists!')

else:

messagebox.showinfo('done',"Booking Added!")

conn.commit()

self.Shows()

conn.close()

def logout(self):

self.destroy()

login()

def onValidate(self, d, i, P, s, S, v, V, W):

try:

int(P)

except ValueError:

return False

else:

return True

def clear(self):

for widget in self.winfo\_children():

widget.destroy()

login()

**#User.py**

import sqlite3

from tkinter import \*

from tkinter import messagebox

class User(Tk):

def start(self):

self.title("Online Hotel Booking")

self.login()

self.mainloop()

def login(self):

self.clear()

LL=Label(self,text = "--Login--")

LL.grid(columnspan=2)

L1=Label(self,text = "User Name : ")

L1.grid(row=1,column=0)

E1=Entry(self)

E1.grid(row=1,column=1)

L2=Label(self,text = "Password : ")

L2.grid(row=2,column=0)

E2=Entry(self,show='\*')

E2.grid(row=2,column=1)

B1=Button(self,text="Login",command = lambda: self.validateLogin(E1.get(),E2.get()))

B1.grid(columnspan=2)

LR=Label(self,text = "--Register--")

LR.grid(columnspan=2)

L3=Label(self,text = "Name : ")

L3.grid(row=5,column=0)

E3=Entry(self)

E3.grid(row=5,column=1)

L4=Label(self,text = "User Name : ")

L4.grid(row=6,column=0)

E4=Entry(self)

E4.grid(row=6,column=1)

L5=Label(self,text = "Password : ")

L5.grid(row=7,column=0)

E5=Entry(self,show='\*')

E5.grid(row=7,column=1)

L6=Label(self,text = "Mobile Number : ")

L6.grid(row=8,column=0)

vcmd = (self.register(self.onValidate),'%d','%i','%P','%s','%S','%v','%V','%W')

E6=Entry(self,validate='key',validatecommand=vcmd)

E6.grid(row=8,column=1)

B2=Button(self,text="Register",command=lambda: self.registerUser(E3.get(),E4.get(),E5.get(),E6.get()))

B2.grid(columnspan=2)

def validateLogin(self,username,password):

conn = sqlite3.connect('database.db')

obs = conn.execute("SELECT password FROM users WHERE username LIKE ?",[username])

dbpass = obs.fetchone()

if dbpass is None :

messagebox.showinfo('error',"Username Not Registered!")

elif password == dbpass[0]:

self.userHome(username)

else:

messagebox.showinfo('error',"Invalid Username or Password!")

conn.close()

def registerUser(self,name,username,password,mblenum):

conn = sqlite3.connect('database.db')

try:

conn.execute("INSERT INTO users VALUES (?,?,?,?);",[name,username,password,mblenum])

except :

messagebox.showinfo('error','UserName Already Exists!')

else:

messagebox.showinfo('done',"Registration Successfull")

conn.commit()

conn.close()

def userHome(self,user):

self.clear()

L=Label(self,text = "Hello "+user)

L.pack()

B1=Button(self,text="View Profile",command=lambda: self.viewProfile(user))

B1.pack()

B2=Button(self,text="Book Tables",command=lambda: self.viewMovies(user))

B2.pack()

B3=Button(self,text="Show Bookings",command=lambda: self.showTickets(user))

B3.pack()

B4=Button(self,text='Logout',command=lambda:self.login())

B4.pack()

def viewMovies(self,user):

self.clear()

B1=Button(self,text='Back',command=lambda:self.userHome(user))

B1.pack()

L=Label(self,text="--Menu--")

L.pack()

conn = sqlite3.connect('database.db')

movies=conn.execute("SELECT movieid, moviename,director, duration FROM movies")

lb = Listbox(self)

movielist=[]

for row in movies:

movielist.append(row)

lb.insert("end", row[1])

lb.pack()

conn.close()

B2=Button(self,text='Next',command=lambda:self.viewShows(user,movielist,lb.curselection()[0]))

B2.pack()

def viewShows(self,user,movielist,opt):

self.clear()

B1=Button(self,text='Back',command=lambda:self.viewMovies(user))

B1.pack()

L1=Label(self,text='Menu name : '+movielist[opt][1])

L1.pack()

L2=Label(self,text='Cuisine By '+movielist[opt][2])

L2.pack()

L3=Label(self,text='Prep time : '+str(movielist[opt][3]))

L3.pack

L=Label(self,text="--Booking--")

L.pack()

conn = sqlite3.connect('database.db')

shows=conn.execute("SELECT showid, showtime,ava\_seats FROM shows WHERE movieid LIKE ?",[movielist[opt][0]])

lb = Listbox(self)

showlist=[]

for row in shows:

showlist.append(row)

lb.insert("end", row[1])

lb.pack()

conn.close()

B2=Button(self,text='Next',command=lambda:self.bookShow(user,showlist,lb.curselection()[0],movielist[opt][1]))

B2.pack()

def bookShow(self,user,showlist,opt,moviename):

self.clear()

L1=Label(self, text='Menu : '+moviename)

L1.grid(columnspan=2)

L2=Label(self,text='Booking : '+showlist[opt][1])

L2.grid(columnspan=2)

L3=Label(self,text='Seats Available :'+str(showlist[opt][2]))

L3.grid(columnspan=2)

var=StringVar(self)

var.set(1)

choices=[1,2,3,4,5,6,7,8,9,10]

L4=Label(self,text='Select Tables : ')

L4.grid(row=4,column=0)

O=OptionMenu(self,var,\*choices)

O.grid(row=4,column=1)

B1=Button(self,text='Confirm & Book',command=lambda:self.book(user,showlist[opt][0],var.get()))

B1.grid(row=5,column=0)

B2=Button(self,text='Cancel',command=lambda:self.userHome(user))

B2.grid(row=5,column=1)

def book(self,user,showid,seats):

conn = sqlite3.connect('database.db')

obj=conn.execute("SELECT max(ticketid) FROM tickets")

ticketid=obj.fetchone()[0]

if ticketid is None:

ticketid=1001

else:

ticketid=ticketid+1

try:

conn.execute("UPDATE shows SET ava\_seats=ava\_seats-? WHERE showid LIKE ?;",[seats,showid])

conn.execute("INSERT INTO tickets VALUES (?,?,?,?);",[ticketid,user,showid,seats])

except Exception as ex:

messagebox.showinfo('error','Seats Unavailable!')

else:

messagebox.showinfo('done',"Booking Successfull")

conn.commit()

conn.close()

self.userHome(user)

def showTickets(self,user):

self.clear()

L=Label(self,text = "Hello "+user)

L.pack()

conn = sqlite3.connect('database.db')

tickets=conn.execute('''SELECT ticketid,moviename,showtime,screenno,seats FROM((tickets

LEFT OUTER JOIN shows ON shows.showid=tickets.showid)

LEFT OUTER JOIN movies ON movies.movieid=shows.movieid)

WHERE tickets.username=?;''',[user])

L2=Label(self,text="--Bookings--")

L2.pack()

for row in tickets:

msg='Order No. : '+str(row[0])+'\n'+str(row[1])+'\n'+str(row[2])+'\nBranch '+str(row[3])+'\nSeats : '+str(row[4])

M=Message( self, text=msg,relief = RAISED )

M.pack()

conn.close()

B=Button(self,text='Back',command=lambda:self.userHome(user))

B.pack()

def viewProfile(self,user):

self.clear()

conn = sqlite3.connect('database.db')

obj=conn.execute('SELECT name, password, mobilenum FROM users WHERE username like ? ;',[user])

profile=obj.fetchone()

L=Label(self,text = "--Profile--")

L.grid(columnspan=2)

L1=Label(self,text = "Name : ")

L1.grid(row=1,column=0)

E1=Entry(self)

E1.insert(END,profile[0])

E1.grid(row=1,column=1)

L2=Label(self,text = "User Name : ")

L2.grid(row=2,column=0)

v=StringVar(self,value=user)

E2=Entry(self,textvariable=v,state=DISABLED )

E2.grid(row=2,column=1)

L3=Label(self,text = "Password : ")

L3.grid(row=3,column=0)

E3=Entry(self,show='\*')

E3.insert(END,profile[1])

E3.grid(row=3,column=1)

L4=Label(self,text = "Mobile Number : ")

L4.grid(row=4,column=0)

vcmd = (self.register(self.onValidate),'%d','%i','%P','%s','%S','%v','%V','%W')

E4=Entry(self,validate='key',validatecommand=vcmd)

E4.insert(END,profile[2])

E4.grid(row=4,column=1)

B2=Button(self,text="Update Profile",command=lambda: self.updateProfile(user,E1.get(),E3.get(),E4.get()))

B2.grid(columnspan=2)

B1=Button(self,text='Cancel',command=lambda:self.userHome(user))

B1.grid(columnspan=2)

def updateProfile(self,username,name,password,mblenum):

conn = sqlite3.connect('database.db')

conn.execute("UPDATE users SET name=?,password=?,mobilenum=? WHERE username LIKE ?;",[name,password,mblenum,username])

messagebox.showinfo('done',"Profile Updated!")

conn.commit()

conn.close()

def onValidate(self, d, i, P, s, S, v, V, W):

try:

int(P)

except ValueError:

return False

else:

return True

def clear(self):

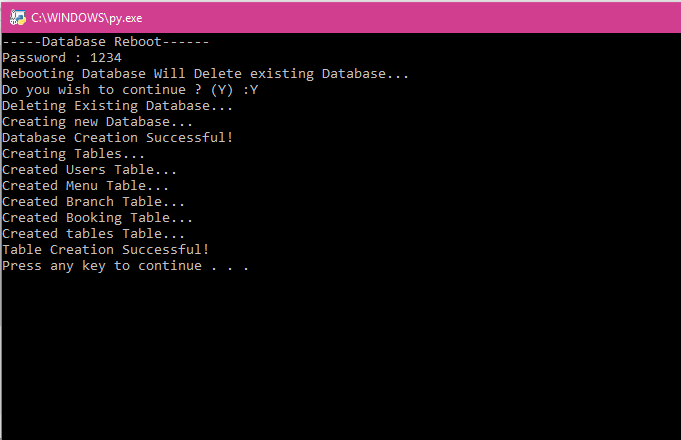
for widget in self.winfo\_children():

widget.destroy()

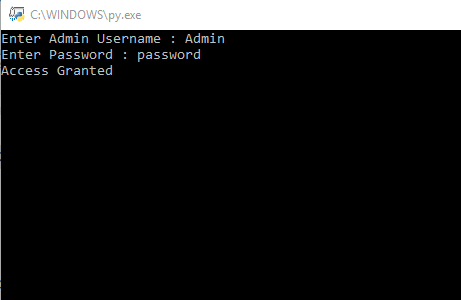
u=User()

u.start()

**SCREENSHOTS :**

Database\_reboot.py

Admin.py



|  |  |
| --- | --- |
| Admin - Home | Admin - Menu |
|  |  |
| Admin - Branch | Admin - Bookings |
|  |  |

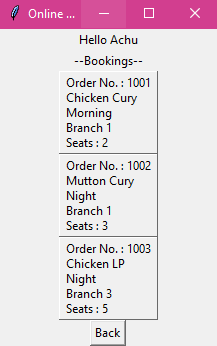
|  |  |
| --- | --- |
| User - Login/Register | User -Registration (Successful) |
|  |  |

|  |  |
| --- | --- |
| User - Login | User - Home |
|  |  |

|  |  |
| --- | --- |
| User - Profile | User- Update |
|  |  |
|  |  |

|  |  |
| --- | --- |
| User - Menu (Menu select) | User - Session (Show selection) |
|  |  |
| User - Booking (Seat selection) | User - Booking (Successful) |
|  |  |

Show booking



**CONCLUSION :**

The application, Hotel Management is a faster, cleaner and a tad more personal website, specially designed to make your booking experience better. Admin was able to add, modify or remove the Menu, branch and booking details.

New users were able to register to the application and registered users were able to update their profile, make bookings and view their Bookings.

**REFERENCES :**

<https://www.tutorialspoint.com/sqlite/sqlite_python.htm>

<https://www.w3schools.com/sql/>

<https://www.google.co.in/>