MOVIE TICKET RESERVATION MANAGEMENT SYSTEM

1. Problem Statement

Develop a database system that contains information about ticket reservations for movie performances. To make a reservation you must be registered as a user of the system. In order to register you choose a unique username and enter your name, address, and telephone number. When you use the system later, you just have to enter your username. In the system, a number of theatres show movies. Each theatre has a name and a number of seats. A movie is described by its name only. A movie may be shown several times, but then during different days. This means that each movie is shown at most once on any day. You can only reserve one ticket at a time to a performance and cannot reserve more tickets than are available at a performance. When you make a reservation, you receive a reservation number that you will use when you pick up the ticket.

Requirement analysis:

We are required to store the personal data of the user like username, phone number, contact address and password for login in ‘login info’ table. We need to store the details about the movies being screened and the details of various theatres available for ticket registration in two respective tables i.e. ‘movies’ and ‘theatre’.

Finally, the information inserted by the user regarding the movie, theatre, date and time of the show are inserting into ‘bookings’ table. The contents of this table are used for reference purpose by the administration and also by the user to use it for convenience in referring to the registration id in it which is required for fetching the hard copy of the ticket from the theatre, in person, which is a pre-requisite.

Software requirement:

Graphical user interface – Python
Database creation and manipulation – SQL*Plus
Database – Oracle

Potential solution:

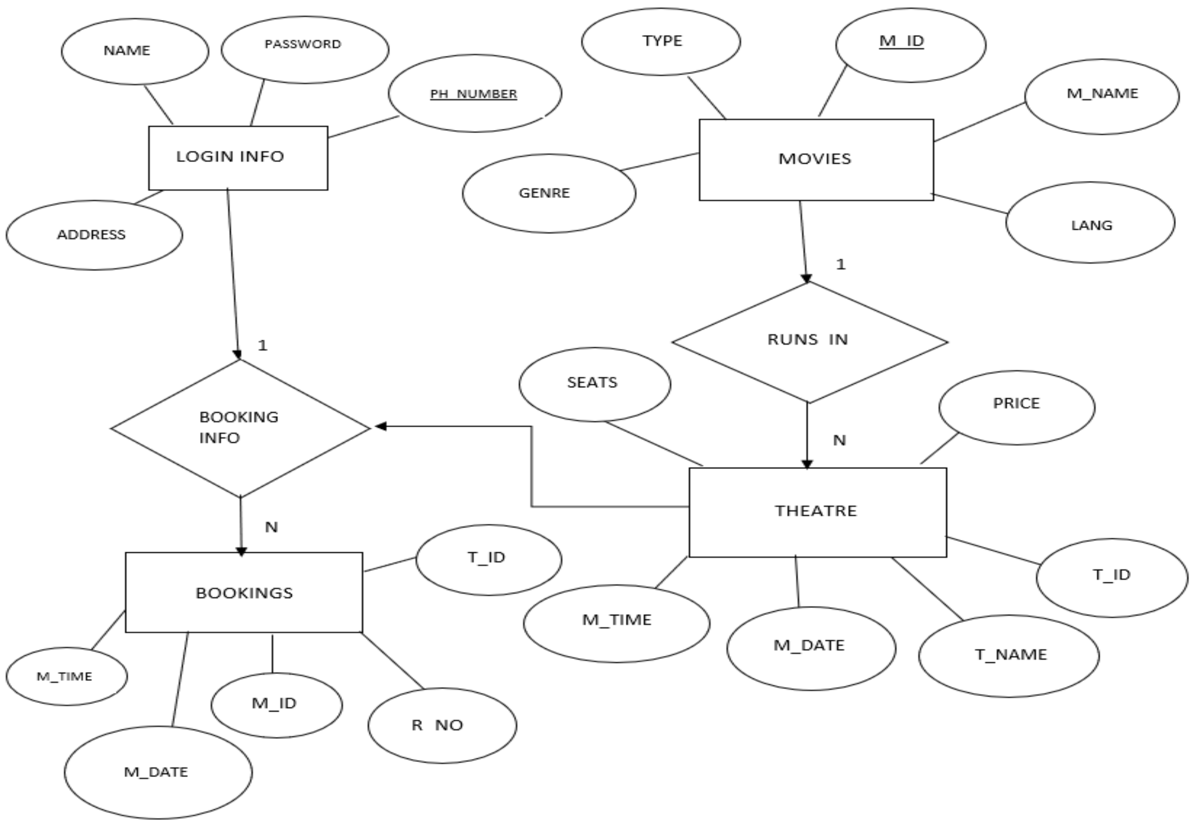
The objective of this project is to develop an efficient movie reservation management system. Initially, the user is provided with the option to either log into the reservation facility or to register himself as a user into the system if it’s his/her first time using the management system. For registration purpose, the user is asked for a username, phone number, contact address and a password and it is stored in ‘login info’ table. The table is referred to authorize login attempts.

Once the user logs in into the system, options to view the movies currently being screened and the theatres in which the movies can be viewed are displayed. Details like genre and type of the movies are displayed to provide the user with a detailed insight of the movies. The theatre information contains information like the availability of seats, price of each ticket, date and time at which a particular movie is to be screened in the respective theatre.

Unique movie id and theatre id are displayed adjacent to the corresponding movies and theatres to make the system more user friendly. The users can type the movie id, theatre id, date and time preferred in the given spaces to reserve a ticket for the movie they want to watch in the theatre they prefer at a particular date and time.

The user is provided with a “Book ticket” button, which when pressed confirms the booking and reserves a ticket for the user. Finally, the ticket detail is displayed along with a unique reference number which can be used by the user for fetching hard copy of the ticket from the respective theatre. The user can then log out of the system or continue booking more tickets.

2. ER Diagram



3. Database Schema

LOGIN INFO TABLE:

Name	Phone_number	Address	Password

MOVIES TABLE:

M_ID	M_Name	Type	Language	Genre

THEATRE TABLE:

T_ID	M_ID	T_Name	M_Date	M_Time	Seats	Price

BOOKINGS TABLE:

Phone_Number	Reg_no	T_ID	M_ID	M_Date	M_Time

4. Database normalization

Normalization is a technique to reduce or remove redundancy from a table.
In this Normalization we have arrived with 4 tables with no redundancy for the database.
The process of arriving at the conclusion is discussed below:

1 NF:

Name	Password	P_number	Address	M_id	M_Name	Type	Language	Genre	T_id	T_Name	M_date	M_time	Seats	Price	R_no

Name	Password	P_number	Address

M_id	M_name	Type	Language	Genre	T_id	T_name	M_date	M_time	Seats	Availability	Price

Table should not contain multivalued attributes for a 1NF configuration. So, the table has been split up in such a way there are no multivalued attribute in the table.

2 NF:

P_number	T_id	M_id	M_time	M_date	R_no

T_id	M_id	M_name	T_Name	Type	Language	Genre	M_time	M_date	Seats	Price

In the 2nd normal form, the table must be in 1NF configuration and all the non-prime attributes should be fully functionally dependent on candidate key.

3 NF:

T_id	M_id	T_name	M_date	M_time	Seats	Price

T_id	T_name

M_id	M_Name	Type	Language	Genre

In the 3rd normal form, the table must be in 2NF and there should be no transitive dependency on the table.
So, you split the table further until you get no redundancy to get the tables
for the required database.

5. Coding & Implementation

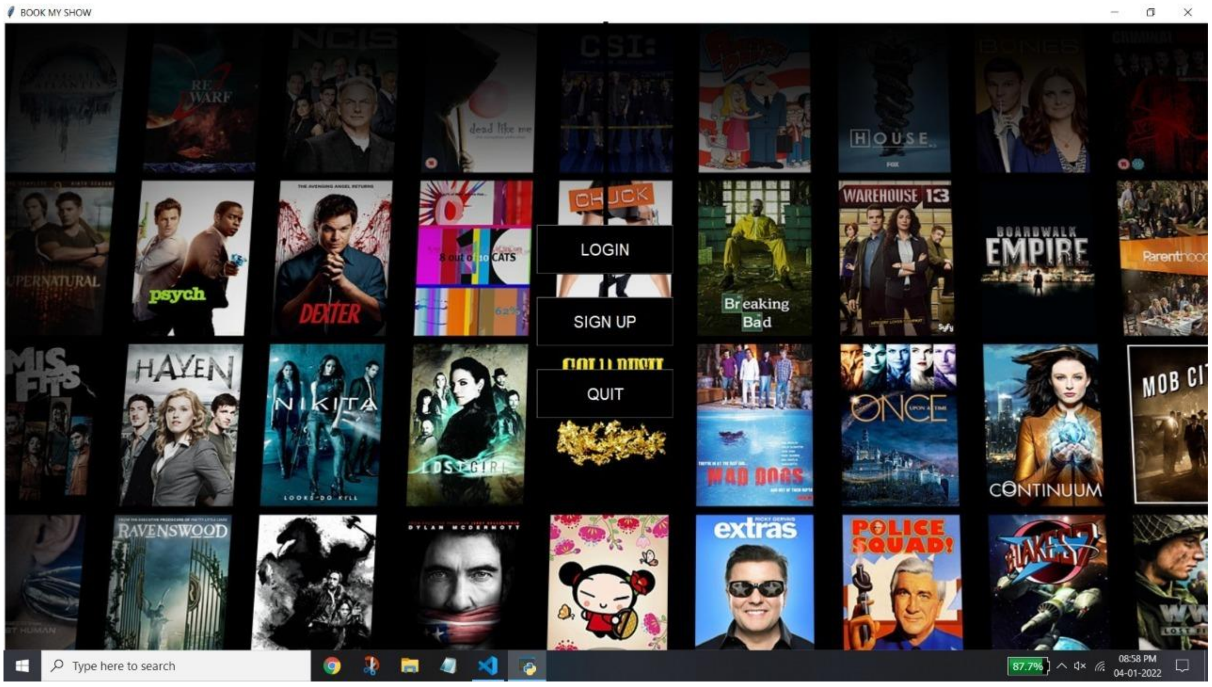
Contents of Project GUI.py:

```
import cx_Oracle
from tkinter import *
from tkinter import messagebox
#########################

con = cx_Oracle.connect('system/nithin@localhost:1521/xe')
#con=cx_Oracle.connect('<userid>/<userpwd>@winsrv:1521/orcl')
cur = con.cursor()
cur2 = con.cursor()

#########################

def home_page(s1):

    def showallmovies():
        lb.delete(0, END)
        for i in cur.execute("select * from movies"):
            lb.insert('end', 'Movie ID - '+ i[0]+ ' | Movie
Name - '+ i[1] + ' ' + i[2])
            lb.insert('end', 'Language - '+ i[3] +' | Genre -
'+ i[4])
            lb.insert('end', '\n')

    def showalltheatres():
        lb.delete(0, END)
        for i in cur.execute("select * from theatres order by
mdate"):
            cur2.execute(f"select name from movies where mid =
'{i[1]}'")
            moviename = cur2.fetchone()[0]
            lb.insert('end', 'Theatre ID - ' + i[0] + ' |
Theatre Name - ' + i[2])
            lb.insert('end', 'Movie ID - '   + i[1] + '    |
Movie Name - ' + moviename)
            lb.insert('end', 'Date & Time - ' +
str(i[3]).split()[0]+ ' '+ str(i[4]))
            lb.insert('end','Available Seats - '+ str(i[5]) +
' | Price - ' + str(i[6]))
            lb.insert('end', '\n')
    def showallbookings():
        lb.delete(0, END)
        for i in cur.execute(f"select * from bookings where
pno = {s1}"):
            lb.insert('end', 'Reservation Number - ' +
str(i[1]) )

            cur2.execute(f"select name from movies where mid =
'{i[3]}'")
            mname = cur2.fetchone()[0]

            cur2.execute(f"select name from theatres where tid
= '{i[2]}'")
            tname = cur2.fetchone()[0]

            lb.insert('end', 'Theatre Name - ' + tname + ' |
Movie Name - ' + mname)
            lb.insert('end', 'Date & Time - '+
str(i[4]).split()[0]+ ' '+ str(i[5]))
            lb.insert('end', '\n')

    def booktickets():
        ss1 = str(se1.get())
        ss2 = str(se2.get())
        ss3 = str(se3.get())
        ss4 = str(se4.get())

        cur.execute(f"select count(*) from theatres where tid
= '{ss2}' and mid = '{ss1}' and mdate = '{ss3}' and mtime =
'{ss4}'")
        number_of_rows = cur.fetchone()[0]

        if number_of_rows == 0:
            messagebox.showwarning(" ","Incorrect Values!")

        else:
          try:
            cur.execute(f"select seats from theatres where tid
= '{ss2}' and mid = '{ss1}' and mdate = '{ss3}' and mtime =
'{ss4}'")
            seats = cur.fetchone()[0]
            if seats > 1:
                cur.execute(f"insert into bookings
values({s1}, rno.nextval, '{ss2}', '{ss1}', '{ss3}',
'{ss4}')")
                cur.execute(f"update theatres set seats =
seats - 1 where tid = '{ss2}' and mid = '{ss1}' and mdate =
'{ss3}' and mtime = '{ss4}' ")
                se1.delete(0, END)
                se2.delete(0, END)
                se3.delete(0, END)
                se4.delete(0, END)
                lb.delete(0, END)
                lb.insert('end', 'Ticket Booked
Successfully!')
            else:
                messagebox.showwarning(" ","Not Enough seats")

          except:
            messagebox.showwarning(" ","No Tickets available
for given record")

        con.commit()

    root = Tk()
    root.geometry('1920x1440')
    root.state('zoomed')
    root.title("BOOK MY SHOW")
    root.option_add( "*font", "Arial" )

    bg = PhotoImage(file =
"C:/Users/Nithin/OneDrive/Desktop/bg2.png")
    imglabel = Label( root, image = bg)
    imglabel.place(x = 0, y = 0)

    cur.execute(f"select username from logininfo where pnumber
= {s1}")
    username = cur.fetchone()[0]

    Label(root, text = f'Welcome Back, {username}!!', fg =
'white', bg = 'black', width = 1920).pack()
    Label(text= '').pack()
    Label(root, text ="MOVIE ID",width=20, fg = 'white', bg =
'black').pack()
    se1 = Entry(root)
    se1.pack()
    Label(root, text ="THEATRE ID", width=20, fg = 'white', bg
= 'black').pack()
    se2 = Entry(root)
    se2.pack()

    Label(root, text ="DATE (Like 01-JAN-22)",width=20, fg =
'white', bg = 'black').pack()
    se3 = Entry(root)
    se3.pack()

    Label(root, text ="TIME in 24 HR Format",width=20, fg =
'white', bg = 'black').pack()
    se4 = Entry(root)
    se4.pack()
    Label(text= '').pack()
    Button(root, text = "BOOK TICKETS", fg = 'white', bg =
'black', width=20, command = booktickets).pack()
    Label(text= '').pack()
    Button(root, text = "SHOW ALL MOVIES",fg = 'white', bg =
'black', width=20, command = showallmovies).pack()
    Button(root, text = "SHOW ALL THEATRES",fg = 'white', bg =
'black', width=20, command = showalltheatres).pack()
    Button(root, text = "SHOW MY BOOKINGS", fg = 'white', bg =
'black', width=20, command = showallbookings).pack()

    Label(text= '').pack()
    Button(root, text = "LOG OUT", fg = 'white', bg = 'black',
width=20, command = lambda:[root.destroy(),
login_page()]).pack()


    #list box################################
    Label(text= '').pack()
    scroll_bar = Scrollbar(root)
    scroll_bar.pack(side = RIGHT, fill = Y )
    lb = Listbox(root, width=50, height=50, yscrollcommand =
scroll_bar.set, fg = 'white', bg = 'black' )
    lb.pack(fill = BOTH )
    scroll_bar.config( command = lb.yview )
    #########################################

    root.mainloop()
```

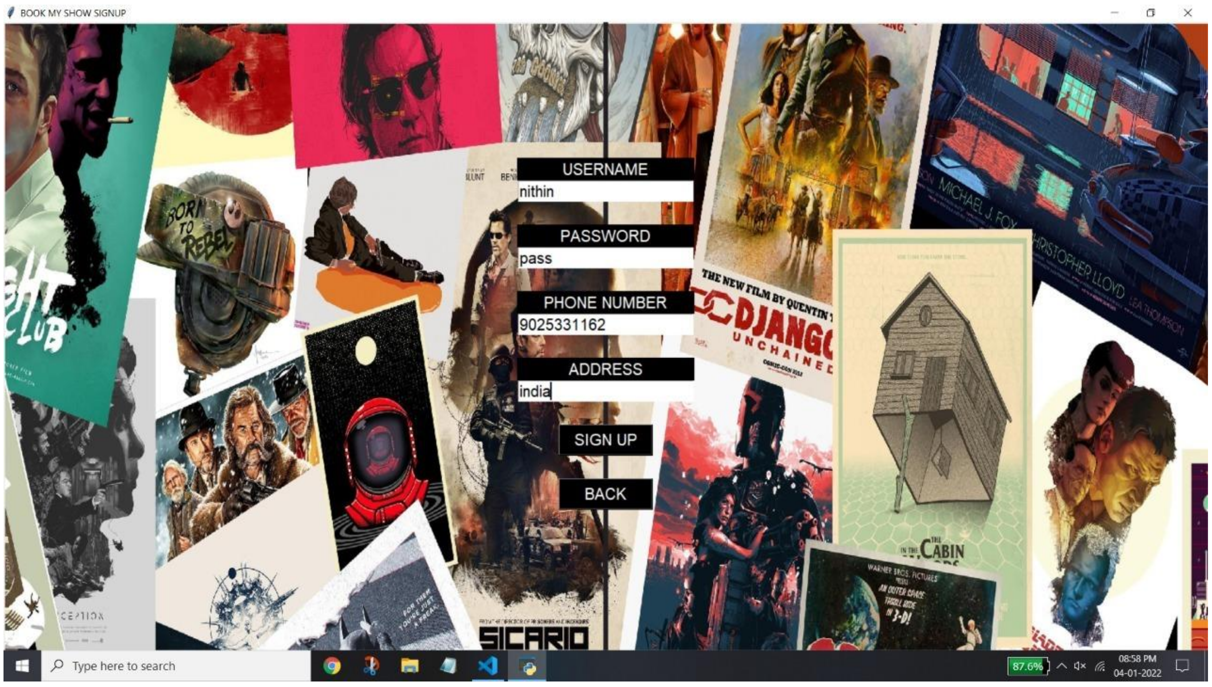
```
##############################################################
############################
def login_page():
    def loginpress():
        s1 = le1.get()
        s2 = str(le2.get())

        cur.execute(f"select count(*) from logininfo where
pnumber = {s1} and password = '{s2}'")
        number_of_rows=cur.fetchone()[0]

        if number_of_rows == 0:
            messagebox.showwarning(" ","Account doesn't
exist")

        else:
            root.destroy()
            home_page(s1)
        con.commit()

    root = Tk()
    root.geometry('1920x1440')
    root.state('zoomed')
    root.title("BOOK MY SHOW LOGIN")
    root.option_add( "*font", "Arial" )

    bg = PhotoImage(file =
"C:/Users/Nithin/OneDrive/Desktop/bg2.png")
    imglabel = Label( root, image = bg)
    imglabel.place(x = 0, y = 0)

    Label(root, text = '\n'*8, bg = '#222024').pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="PHONE NUMBER", fg = 'white', bg =
'black', width = 20).pack()
    le1 = Entry(root)
    le1.pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="PASSWORD", fg = 'white', bg = 'black',
width = 20).pack()
    le2 = Entry(root)
    le2.pack()
    Label(text= '', bg = '#222024').pack()
    Button(root, text = "LOGIN", fg = 'white', bg = 'black',
height=1, width=20,command = loginpress).pack()

    Label(text= '', bg = '#222024').pack()
    Button(root, text = "CREATE ACCOUNT", fg = 'white', bg =
'black', height=1, width=20, command = lambda:[root.destroy(),
signup_page()]).pack()

    Label(text= '', bg = '#222024').pack()
    Button(root, text = "QUIT", fg = 'white', bg = 'black',
height=1, width=20, command = root.destroy).pack()

    Label(root, text = '\n'*20, bg = '#222024').pack()

    root.mainloop()

##############################################################
###################################

def signup_page():

    def signuppress():
        s1 = str(se1.get())
        s2 = str(se2.get())
        s3 = se3.get()
        s4 = str(se4.get())

        try:
            cur.execute(f"insert into logininfo values('{s1}',
'{s2}', {s3}, '{s4}')")
            se1.delete(0, END)
            se2.delete(0, END)
            se3.delete(0, END)
            se4.delete(0, END)
            messagebox.showinfo(" ", "Signed Up
Successfully\nYou can now login in")
            root.destroy()
            login_page()

        except:
            messagebox.showwarning(" ","Values entered are
either not unique or empty\nNOTE: Phone Number must be
unique")

        con.commit()


    root = Tk()
    root.geometry('1920x1440')
    root.state('zoomed')
    root.title("BOOK MY SHOW SIGNUP")
    root.option_add( "*font", "Arial" )

    bg = PhotoImage(file =
"C:/Users/Nithin/OneDrive/Desktop/bg2.png")
    imglabel = Label( root, image = bg)
    imglabel.place(x = 0, y = 0)

    Label(root, text = '\n'*5, bg = '#222024').pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="USERNAME", fg = 'white', bg = 'black',
width = 20).pack()
    se1 = Entry(root)
    se1.pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="PASSWORD", fg = 'white', bg = 'black',
width = 20).pack()
    se2 = Entry(root)
    se2.pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="PHONE NUMBER", fg = 'white', bg =
'black', width = 20).pack()
    se3 = Entry(root)
    se3.pack()

    Label(text= '', bg = '#222024').pack()
    Label(root, text ="ADDRESS", fg = 'white', bg = 'black',
width = 20).pack()
    se4 = Entry(root)
    se4.pack()
    Label(text= '', bg = '#222024').pack()
    Button(root, text = "SIGN UP", fg = 'white', bg = 'black',
height=1, width=10, command = signuppress).pack()

    Label(text= '', bg = '#222024').pack()
    Button(root, text = "BACK", fg = 'white', bg = 'black',
height=1, width=10, command = lambda:[root.destroy(),
login_page()]).pack()

    Label(root, text = '\n'*20, bg = '#222024').pack()
    root.mainloop()

###########################################

root = Tk()
root.geometry('1920x1440')
root.state('zoomed')
root.title("BOOK MY SHOW")
root.option_add( "*font", "Arial")

bg = PhotoImage(file =
"C:/Users/Nithin/OneDrive/Desktop/bg1.png")
imglabel = Label( root, image = bg)
imglabel.place(x = 0, y = 0)

Label(text= '\n'*10, bg = 'black').pack()
Button(root, text="LOGIN", height=2, width=15, bg = 'black',
fg='white', command = lambda:[root.destroy(),
login_page()]).pack()

Label(text= '', bg = 'black').pack()
Button(root, text="SIGN UP", height=2, width=15, bg = 'black',
fg='white',command = lambda:[root.destroy(),
signup_page()]).pack()

Label(text= '', bg = 'black').pack()
Button(root, text="QUIT", height=2, width=15, bg = 'black',
fg='white',command = root.destroy).pack()

root.mainloop()
```

6. Sample Input/Outputs

Opening page



Sign up page for new user with sample entries:


BOOK MY SHOW SIGNUP

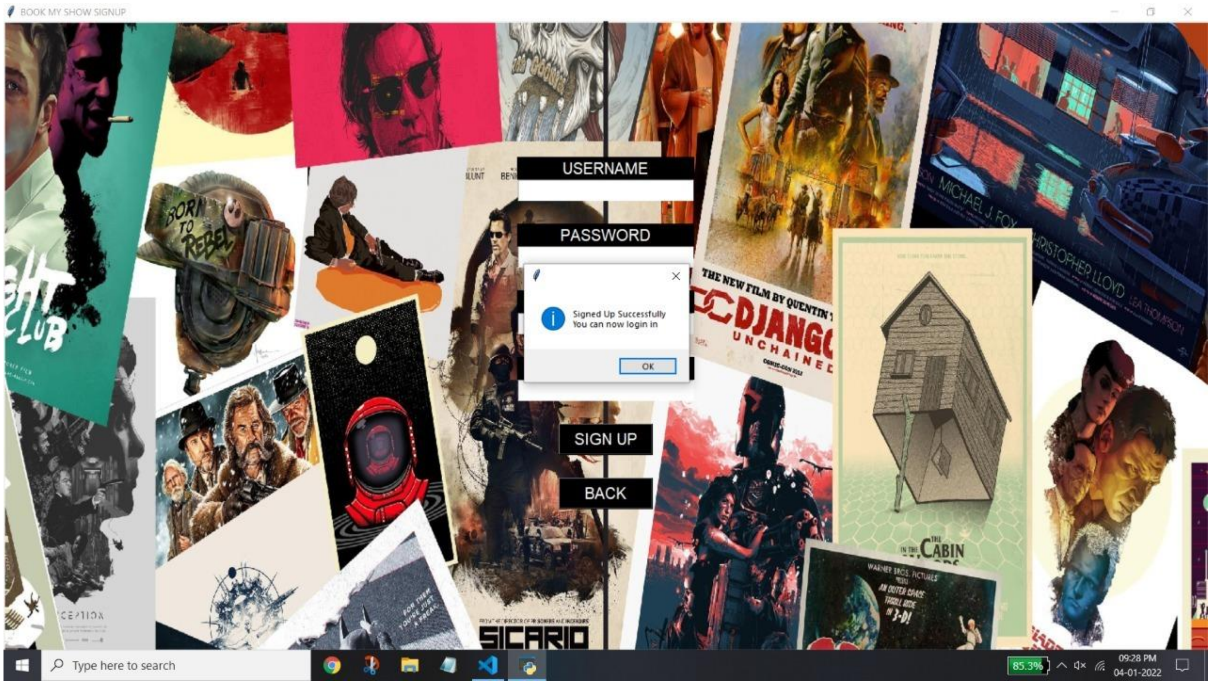
USERNAME
nithin

PASSWORD
pass

PHONE NUMBER
9025331162

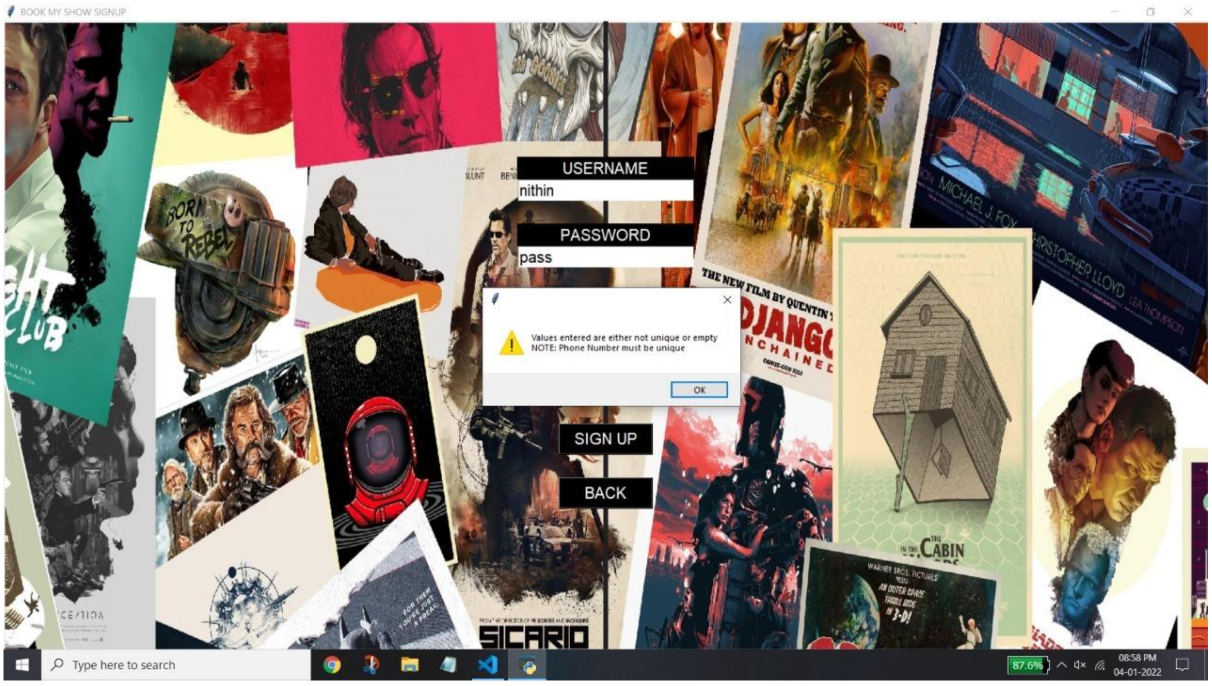
ADDRESS
india

Successful sign up:



Signed Up Successfully
You can now login in

Invalid sign up:

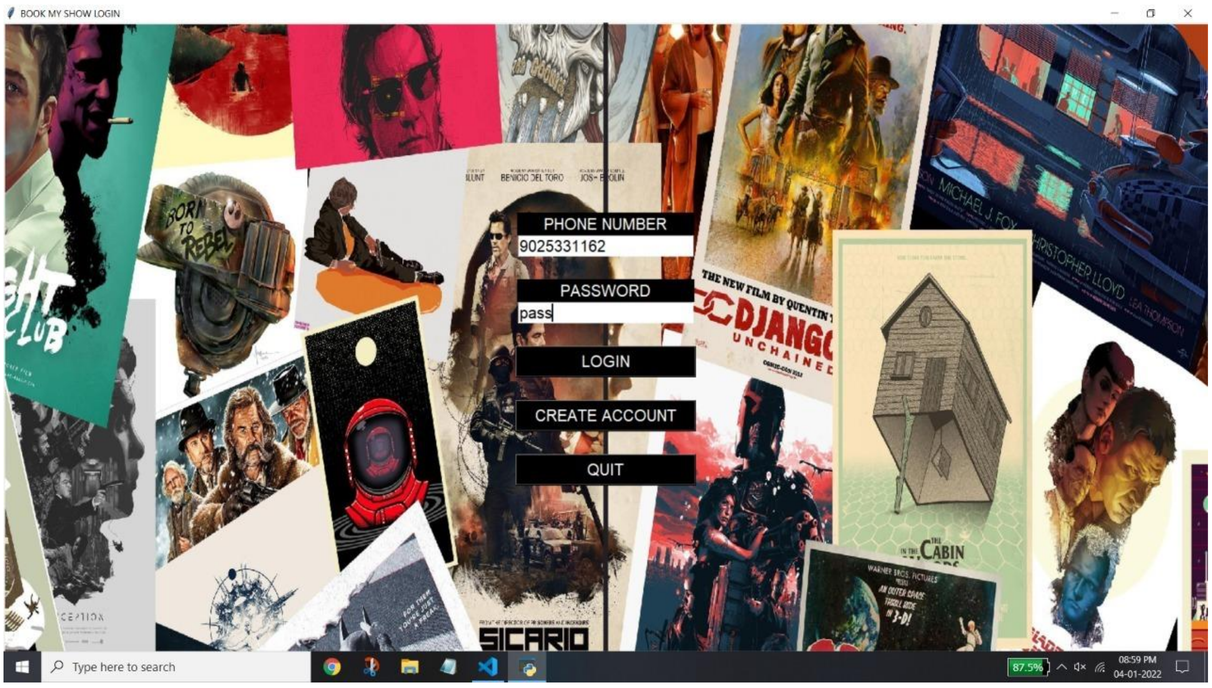


nithin

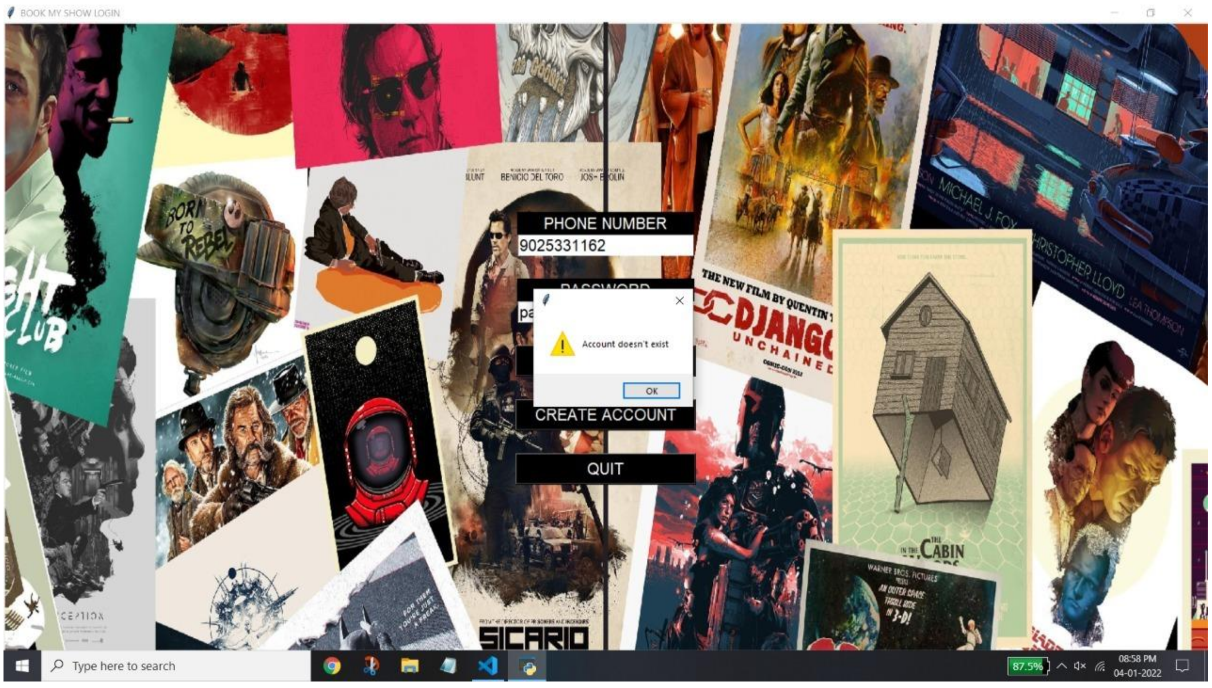
pass

Values entered are either not unique or empty
NOTE: Phone Number must be unique

Login page with sample entries:

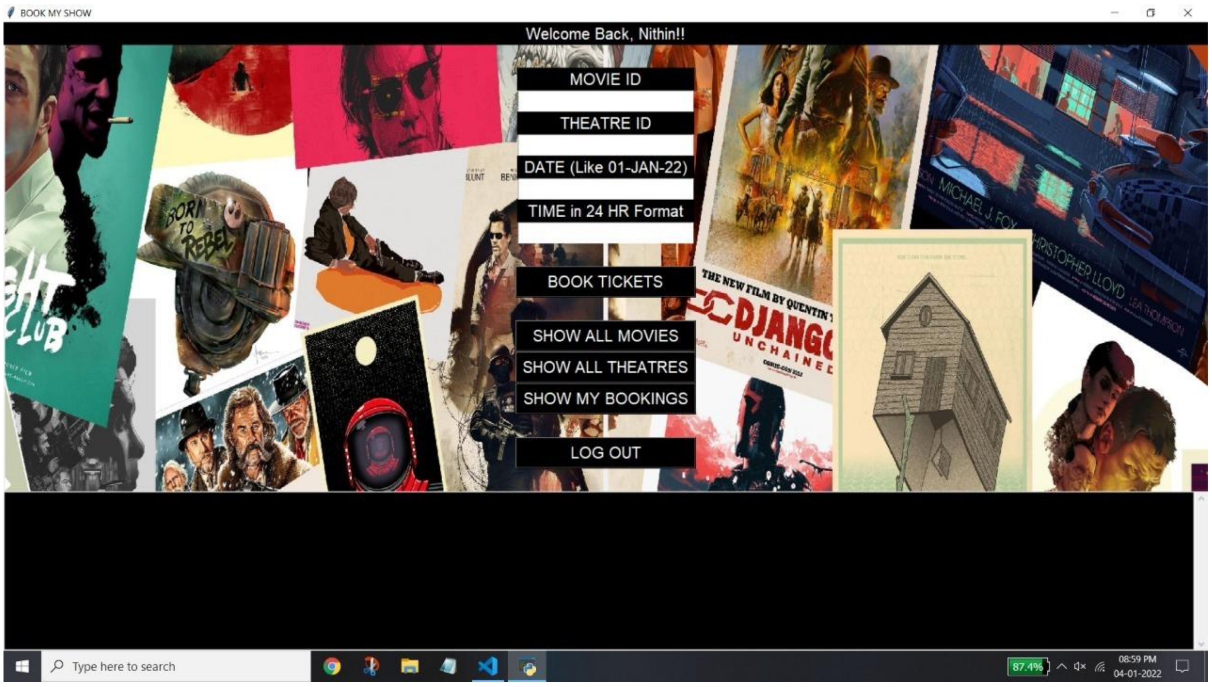


Invalid login:

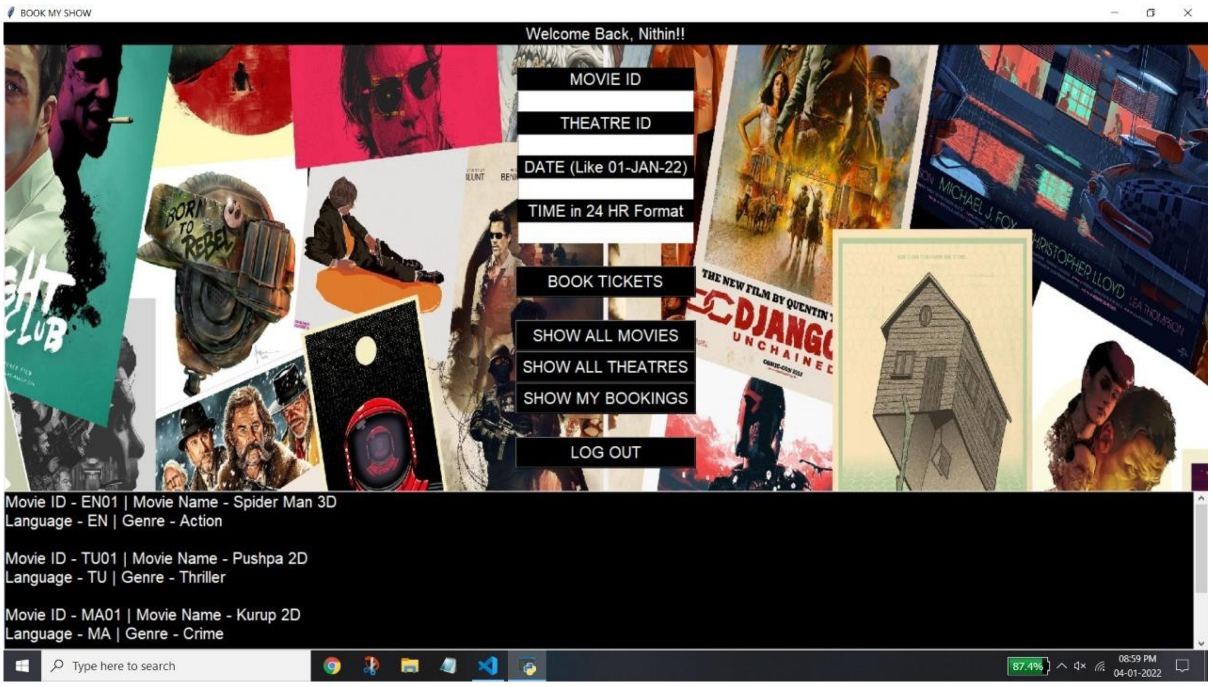


Account doesn't exist

Successful login results in opening of home page of the system with a welcome message to the user:

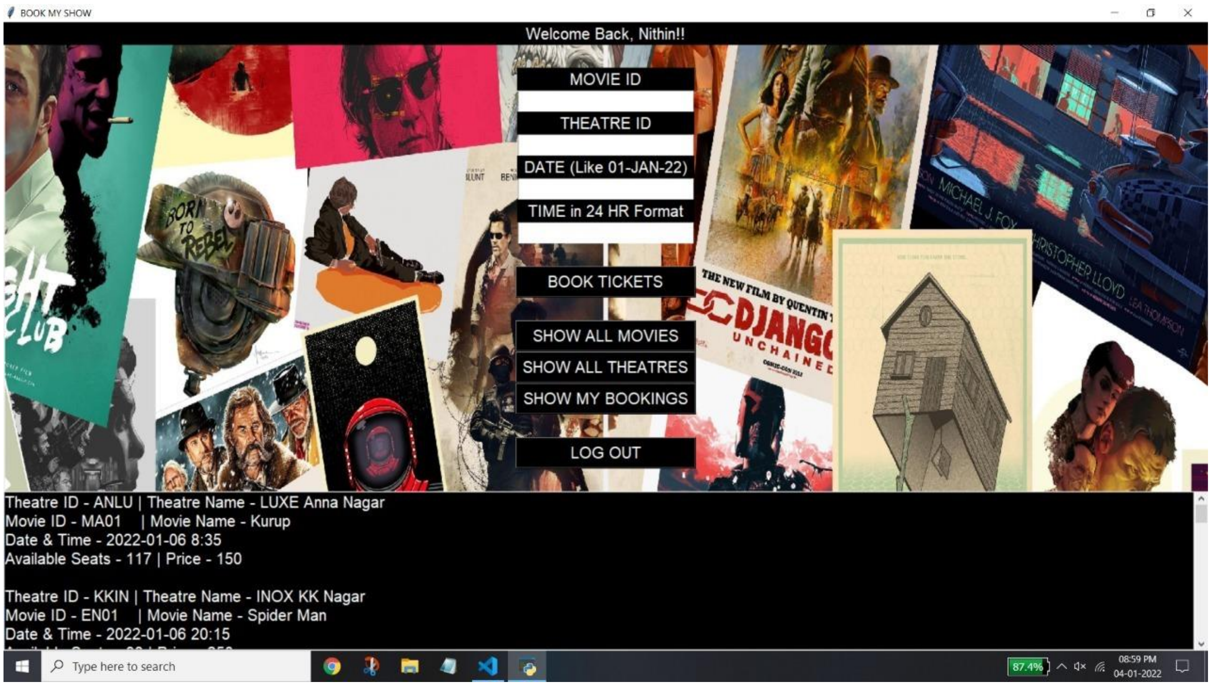


Functioning of "SHOW ALL MOVIES" button demonstrated:



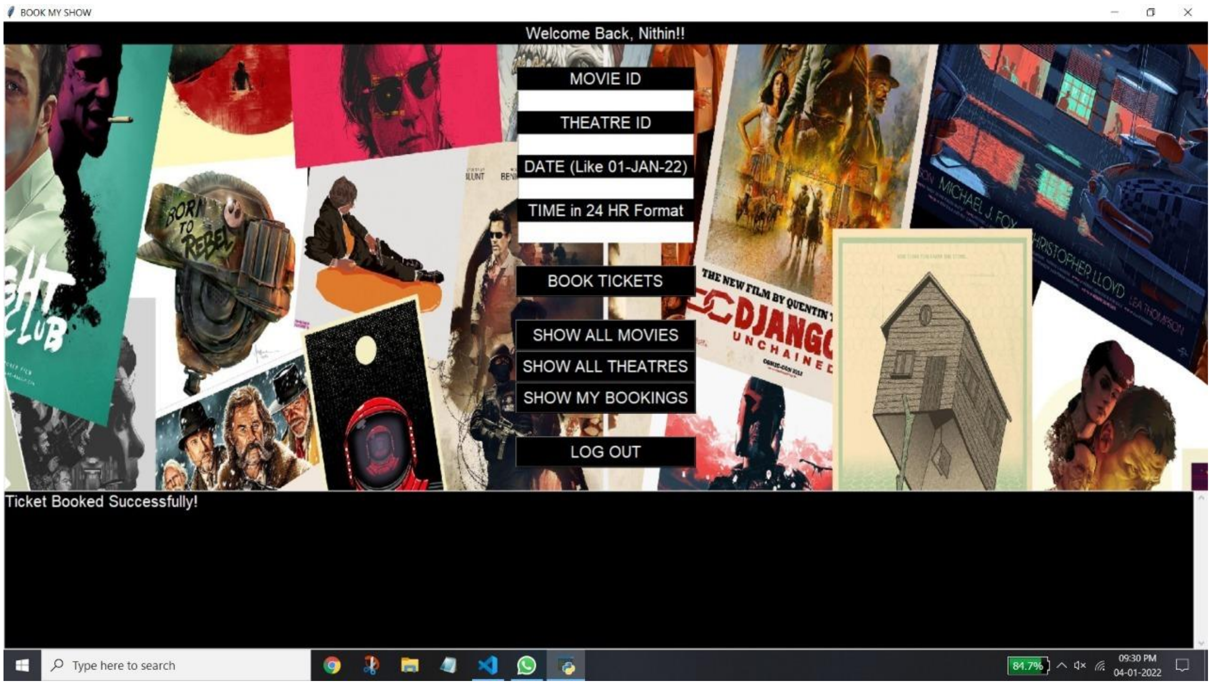
```
Movie ID - EN01 | Movie Name - Spider Man 3D
Language - EN | Genre - Action

Movie ID - TU01 | Movie Name - Pushpa 2D
Language - TU | Genre - Thriller

Movie ID - MA01 | Movie Name - Kurup 2D
Language - MA | Genre - Crime
```

Functioning of “SHOW ALL THEATRES” button demonstrated:



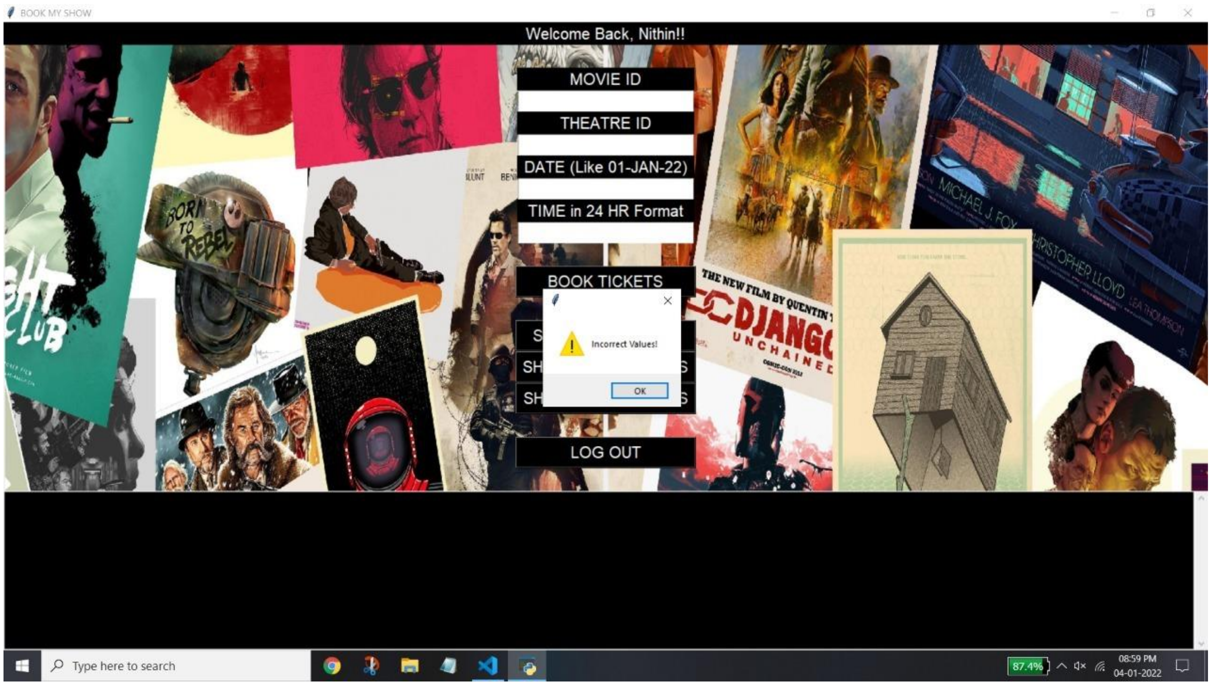
```
Theatre ID - ANLU | Theatre Name - LUXE Anna Nagar
Movie ID - MA01    | Movie Name - Kurup
Date & Time - 2022-01-06 8:35
Available Seats - 117 | Price - 150

Theatre ID - KKIN | Theatre Name - INOX KK Nagar
Movie ID - EN01    | Movie Name - Spider Man
Date & Time - 2022-01-06 20:15
```

Successful booking message:

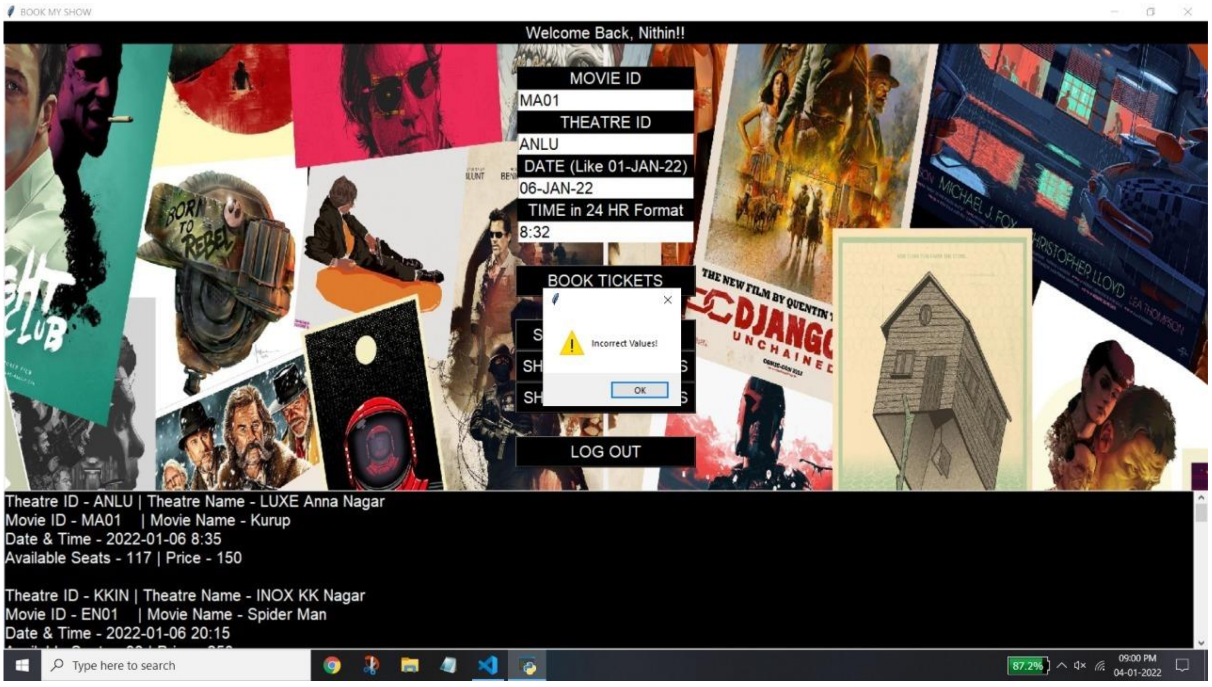


```
Ticket Booked Successfully!
```

Invalid ticket booking (The entry boxes are not filled):



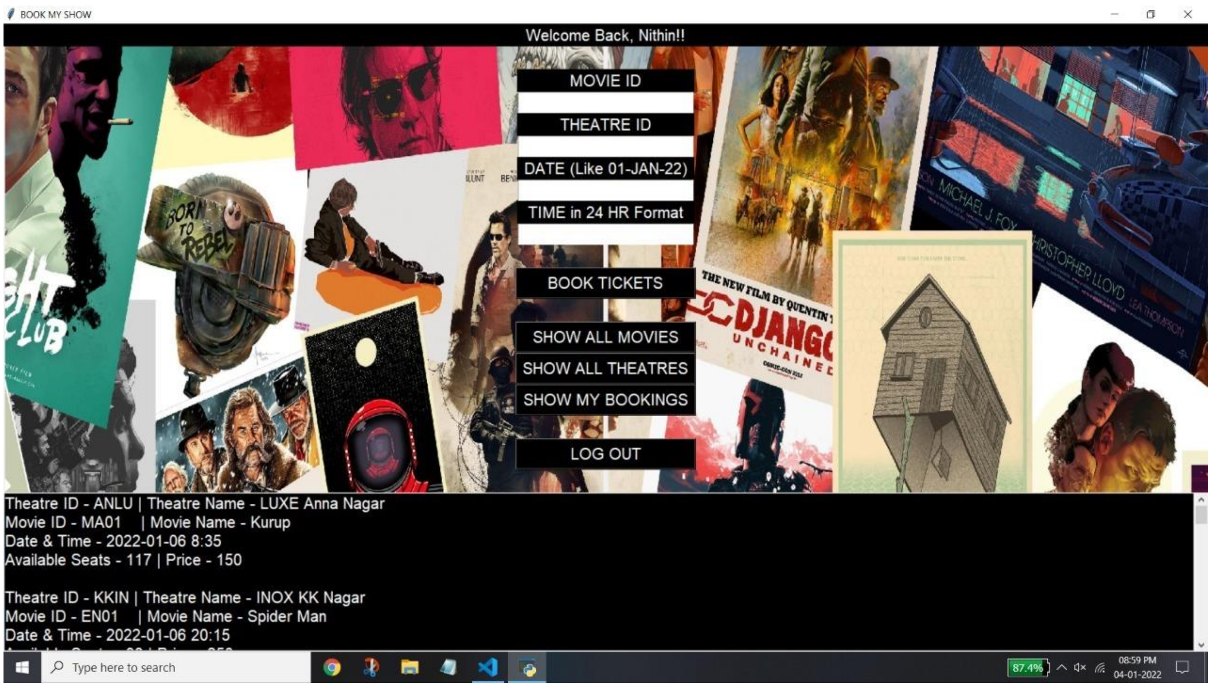
Invalid ticket booking (The entry boxes are filled with incorrect information):



```
Theatre ID - ANLU | Theatre Name - LUXE Anna Nagar
Movie ID - MA01    | Movie Name - Kurup
Date & Time - 2022-01-06 8:35
Available Seats - 117 | Price - 150

Theatre ID - KKIN | Theatre Name - INOX KK Nagar
Movie ID - EN01    | Movie Name - Spider Man
Date & Time - 2022-01-06 20:15
```

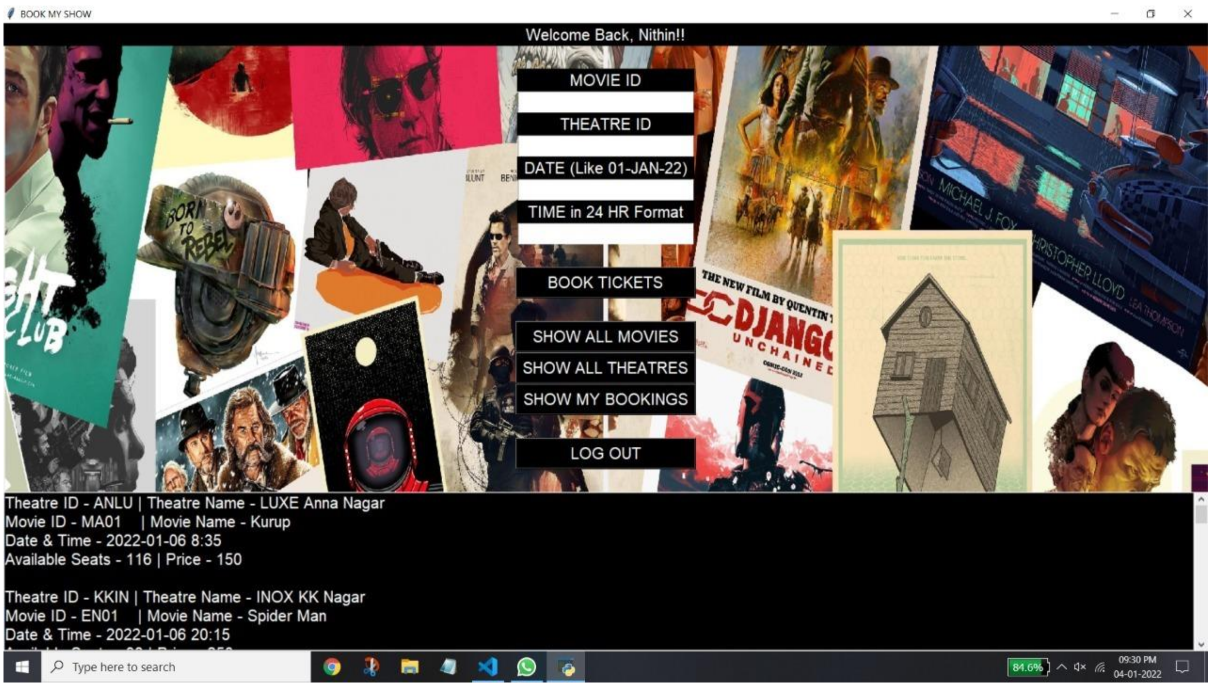
Number of seats getting reduced in the respective theatre for a particular movie after a successful booking:

BEFORE BOOKING



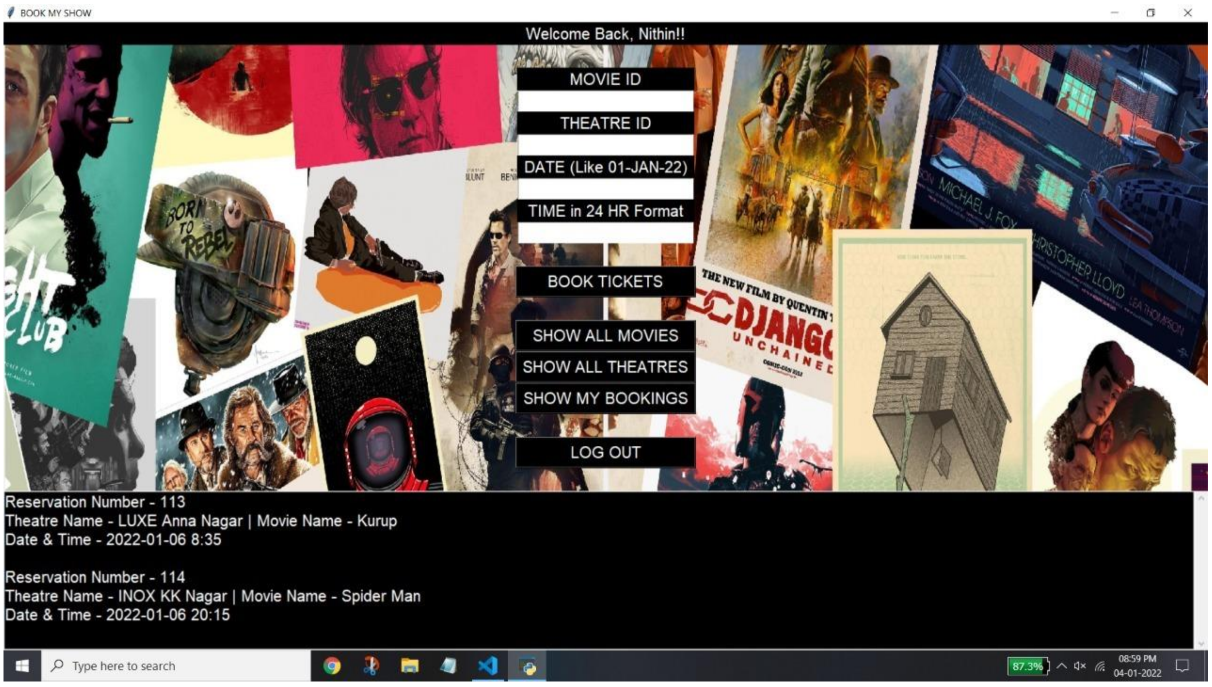
```
Theatre ID - ANLU | Theatre Name - LUXE Anna Nagar
Movie ID - MA01    | Movie Name - Kurup
Date & Time - 2022-01-06 8:35
Available Seats - 117 | Price - 150

Theatre ID - KKIN | Theatre Name - INOX KK Nagar
Movie ID - EN01    | Movie Name - Spider Man
Date & Time - 2022-01-06 20:15
```

AFTER BOOKING



```
Theatre ID - ANLU | Theatre Name - LUXE Anna Nagar
Movie ID - MA01    | Movie Name - Kurup
Date & Time - 2022-01-06 8:35
Available Seats - 116 | Price - 150

Theatre ID - KKIN | Theatre Name - INOX KK Nagar
Movie ID - EN01    | Movie Name - Spider Man
Date & Time - 2022-01-06 20:15
```

Showing all booked tickets:



```
Reservation Number - 113
Theatre Name - LUXE Anna Nagar | Movie Name - Kurup
Date & Time - 2022-01-06 8:35

Reservation Number - 114
Theatre Name - INOX KK Nagar | Movie Name - Spider Man
Date & Time - 2022-01-06 20:15
```

Appendix-Report on Python GUI tkinter

Tkinter is the standard GUI library for Python. It is the Python interface to the Tk GUI toolkit shipped with Python.

To create a window in tkinter we user a Tk() object, which allows us to create a standalone window.

```
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

The above code creates a blank window when executed.

Tkinter Widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Brief description of all the Tkinter widgets used in this project.

1. Label:

Label widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time.

Syntax:

l = Label (master, option, ...)

Parameters:

- ***master*** − This represents the parent window.
- ***options*** – text, font, bg, image, justify, fg, bitmap etc.

2. Button:

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Syntax:

b = Button (master, option=value, ...)

Parameters:

- ***master*** − This represents the parent window.

- ***options*** – text, font, bg, fg, image, height, width, command etc.

3. Scrollbar:

This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text and Canvas.

Syntax:

s = Scrollbar (master, option, ...)

Parameters:

- ***master*** − This represents the parent window.
- ***options*** – bg, bd, command, orient, width, repeatinterval, takefocus etc.

4. Toplevel:

Toplevel widgets work as windows that are directly managed by the window manager. They do not necessarily have a parent widget on top of them.

Syntax:

t = Toplevel (master, option, ...)

Parameters:

- ***master*** − This represents the parent window.
- ***options*** – bd, bg, cursor, height, width, relief, font, fg etc.